Title
Security and Performance Considerations in Wireless Networks

Author
Pelechrinis, Konstantinos

Publication Date
2010

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA
RIVERSIDE

Security and Performance Considerations in Wireless Networks

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

by

Konstantinos Pelechrinis

August 2010

Dissertation Committee:

Dr. Srikanth Krishnamurthy, Chairperson
Dr. Michalis Faloutsos
Dr. Walid Najjar

The Dissertation of Konstantinos Pelechrinis is approved by:

_____

_____

_____

Committee Chairperson

University of California, Riverside

ACKNOWLEDGMENTS

I would like to express my gratitude to all those who made this dissertation possible. First of all, I would like to thank my Ph.D advisor, Dr. Srikanth Krishnamurthy. His dedication and enthusiasm in research has greatly inspired me, while his personal contact and advices to me have been priceless, helping me build a strong character. I would also like to thank Dr. Michalis Faloutsos for the academic support and the valuable knowledge I acquired from him during our interactions. His social support was also important to getting over difficulties due to my move to the US. I would like to thank Dr. Walid Najjar, for participating in the committee for my defense. In short, I was very fortunate to have had such a supportive committee, in which each member played a special and indispensable role in my entire graduate career. In addition, I wish to thank Dr. Stephan Eidenbenz, for believing in me during my first steps towards my PhD and collaborating with me. Our collaboration was more than successful and the experience I gained priceless as well. Moreover, I would like to express my gratitude to Dr. Christos Gkantsidis for his support during my stay in Cambridge, UK. I learnt many things interacting with him and his advices have definitely improved me as a scientist and a person. I would especially like to thank my collegue and friend, Marios Kokkodis, for continuously supporting me in the difficulties I faced throughout my studies due to my relocation from Greece to the United States. Finally, I would like to thank my friends and lab-mates, who provided help and made the long journey a lot more enjoyable and memorable: Dr. Ioannis Broustis, Dr. Theodoros Salonidis, Dr. Guanhua Yan, Dr. Iordanis Koutsopoulos, Dr. Christos Koufogiannakis, Dr. Anastasios Mourikis, Dr. Ioannis Drougas, Achilleas Arvanitis, Michalis Raptis, Panagiotis Papadimitriou, Theodoros Lappas, Marios Iliofotou, Aggelos Vlavianos, Aggelos Lazaris, Marco Di Molfetta and many more.

To my family

ABSTRACT OF THE DISSERTATION

Security and Performance Considerations in Wireless Networks

by

Konstantinos Pelechrinis

Doctor of Philosophy, Graduate Program in Computer Science
University of California, Riverside, August 2010
Dr. Srikanth Krishnamurthy, Chairperson

The open and shared nature of the wireless medium makes it easy for adversaries to launch simple, yet effective, denial of service attacks (DoS attacks). As an example, jamming attacks, involve the uncoordinated transmission of electromagnetic energy on the medium. In a carrier sensing network (e.g., 802.11), this attack strategy increases the number of collisions at the receiver side and/or blocks the medium access to legitimate nodes at the transmitting side. Both of the above effects degrade the wireless network performance significantly. Frequency hopping (FH) has been traditionally used to overcome jamming attacks. However, we analytically and experimentally show that FH is inadequate to efficiently cope with jamming in today's networks.

Later we propose a suite of systems that aim at coping with jamming attacks at various levels (i.e., detection, localization and prevention). We first identify two intelligent and effective jamming attacks that can be launched in 802.11 WLANs and we provide robust detection systems. In particular, we design and implement (i) CMD, a system to detect *active* jamming attacks that exploit the carrier sensing functionality of 802.11 networks and (ii) FIJI, a cross-layer system for detecting (and mitigating) jamming attacks that exploit the performance anomaly of 802.11 WLANs. Furthermore, given the importance of locating the jamming device in many deployment scenarios (e.g., battlefield), we propose a lightweight jamming localization scheme. Our system utilizies

ideas borrowed from the gradient descent optimization method. The system's evaluations, show the potentials and applicability of our localization strategy. The final step for coping with jamming attacks is *jamming prevention*. Based on our initial measurement driven analysis, we do not rely on a FH scheme, that tries to simply avoid the jammer. On the contrary, we design, implement and evaluate a prevention system, called ARES (Anti-jamming REinforcement System), to *fight* against the saboteur. ARES is applicable to carrier sensing networks and tunes the parameters of rate adaptation and power control to improve the performance under the presence of an attack while ensuring that operations under benign conditions are unaffected. Our extensive evaluations, show that ARES improves the network throughput across all scenarios by up to 150%.

Contents

List of Figures

Chapter 1

Introduction

The nature of wireless networks makes them more vulnerable to DoS attacks as compared to traditional wireline networks. Wireless communications take place over the air, and thus, are prone to a variety of malicious strategies. As a simple example, an eavesdropper can use commodity hardware to capture packets transmitted between two wireless nodes. If encryption is not enforced, or if the scheme used is weak, important information can be revealed for the undergoing communication.

In this thesis we will focus on **jamming attacks** in 802.11 networks. We present a complete suite of protocols that address the **(i)** detection, **(ii)** localization and **(iii)** mitigation of this kind of attacks.

1.1 The Jamming Attack

A jammer transmits electromagnetic energy to hinder legitimate communications on the wireless medium. A jamming attack can cause the following effects in an 802.11 (in general a carrier sensing) network: **(a)** Due to carrier sensing, co-channel transmitters defer their packet transmissions for prolonged periods. **(b)** The jamming signal collides with legitimate packets at receivers. Figure 1.1 depicts the operations and effects of a jamming node. Frequency hopping techniques have been previously proposed for avoiding jammers [1] [2]. Such schemes however, are not effective

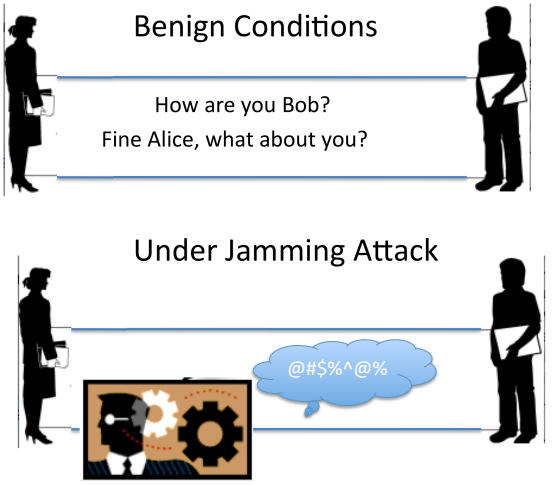


Figure 1.1: Pictorial representation of a jammer.

in scenarios with wide-band jammers [3, 4]. Furthermore, given that 802.11 operates on relatively few frequency channels, multiple jamming devices operating on different channels can significantly hurt performance in spite of using frequency hopping [5].

Jammers can be distinguished in terms of their attack strategy. While a detailed discussion can be found in [6], the traditional jamming models can be classified as:

Non-stop jamming: *Constant* jammers continuously emit electromagnetic energy on a channel. Nowadays, constant jammers are commercially available and easy to obtain [7, 3]. While constant jammers emit non-decipherable messages, *deceptive* jammers transmit seemingly legitimate back-to-back dummy data packets. Hence, they can mislead other nodes and monitoring systems into believing that legitimate traffic is being sent.

Intermittent Jamming: As the name suggests, these jammers are active intermittently; the primary goal is to conserve battery life. A *random* jammer typically alternates between uniformly-distributed jamming and sleeping periods; it jams for T_j seconds and then it sleeps for T_s seconds. A *reactive* jammer starts emitting energy only if it detects traffic on the medium. This makes the jammer difficult to detect. However, implementing reactive jammers can be a challenge.

Attackers are motivated into using a random jammer because putting the jammer to sleep inter-

mittently can increase its lifetime and decrease the probability of detection [6]. Furthermore, it is the most generalized representation of a jammer; appropriately choosing the sleep times could turn the jammer into a constant jammer or (with high probability) a reactive jammer. Moreover, reactive jammers are not easily available since they are harder to implement and require special expertise on the part of the attacker.

1.2 Jamming Countermeasures

When a jamming attack is launched, the network operator needs to have a way to restore the benign operations. Therefore, a system for **mitigating** the attack's effects is required. Based on the details of the design and/or implementation of the system, the network performance might be significantly affected under benign settings if the system runs continuously. As an example, FH involves frequent switches among the different available channels. Based on the hardware and software used, these hops might require a non-negligible time period to happen, during which there is no communication taking place. If we add up synchronization delays, it is easy to see that a carelessly deployed FH system can significantly affect the network perforance under benign conditions. Therefore, prior to applying any alleviation technique, a jamming **detection** scheme should be deployed which will eventually trigger the usage of the prevention system whenever needed.

In addition, under the presence of an extremely powerful jammer (both in terms of power and frequency), most mitigation schemes will fail to alleviate the jamming effects. Thus, the ability to find the exact location of the malicious device becomes imperative. This can augment network operations by avoiding areas that are in the vicinity of the jamming node. As an example, once the location of the saboteur is identified routing functionalities can incorporate this information and avoid routes passing near the jammer. In addition, spatial evasion techniques can be bootstrapped, while the physical deactivation of the adversarial device will become viable. Anti-jamming research spans all of the above areas and tries to efficiently and effectively address jammer detection, localization and mitigation.

1.3 Contributions

As alluded to above, a full-fledged anti-jamming system should include functionalities for detection and mitigation of jamming attacks as well as localization of the malicious device. Previous studies, discussed in details in each chapter, propose solutions that are focused on a specific jamming sub-problem. Furthermore, usually the applicability of these solutions is limited to specific jamming models/strategies, as it will be evident in the following. Having the objective of a complete anti-jamming solution, the main contributions of this thesis can be summarized in the following:

- We analytically and experimentally show, that frequency hopping is inadequate for coping with jamming attacks in current 802.11 networks [5]. We design a generic measurement-driven, game theoretic framework that captures the interactions between malicious node(s) and a communication link that employ frequency hopping. Applying our model to the case of 802.11 networks we find that with current frequency allocation, 5 jammers can complete break down the communications that take place on all available orthogonal channels. This result sets FH a rather weak countermeasure against jamming attacks at 802.11 networks.

- We identify two intelligent and efficient jamming attacks that can significantly degrade the performance of WLANs and cannot be detected by previously proposed schemes [2] [6] [8]. The first attack, which we call *active jamming*, incorporates the tuning of the Clear Channel Assesment threshold (CCA) with the objective to gain more access to the medium while starving the legitimate clients with respect to the available bandwidth. We propose a challenge - response detection scheme, called CMD, which achieves high detection accuracy with low false positives [9]. In addition, we identify an *implicit jamming* attack, which takes advantage of the performance anomaly of 802.11 WLANs [10]. In particular, the placement of a low-power jammer in a way that it affects a single legitimate client can cause starvation to all the other clients [11]. We design and implement a cross layer detection (and mitigation) system called FIJI; FIJI looks for anomalies in the AP load distribution to efficiently perform jammer detection. In addition, it includes a traffic shaping module to mitigate the implicit

jamming attack.

- We propose a lightweight jammer localization scheme based on the gradient descent optimization algorithm [12]. The key observation that drives the design of our system is the fact that the Packet Delivery Ratio (PDR) exhibits its minimum value at the areas around the jamming device; the corresponding value increases as we move further from the adversarial's device. We modify the gradient descent method to be applicable to the discrete plane of the network topology and to design and implement our localization system. Our testbed evaluations reveal the potentials of the proposed approach.

- We design and implement a measurement-driven anti-jamming system, called ARES [13]. Our measurements indicate that **(a)** the use of popular rate adaptation algorithms can significantly degrade the networks performance, and **(b)** appropriate tuning of the carrier sensing threshold allows a transmitter to send packets even when being jammed and enables receiver *capture* the desired signal. ARES tunes the parameters of the aforementioned physical layer functionalities to improve the performance under the presence of jammers, while ensuring unaffected operations under benign settings. Our evaluations across a variety of settings, show improvements in network throughput by up to 150%.

1.4 Outline of this dissertation

The rest of this dissertation is organized as follows. In chapter 2, we provide our measurement driven, game theoretic framework that assesses the efficiency of frequency hopping in mitigating jamming attacks. We model the interactions between the jammer and the communication link as a two player, zero-sum game, whose payoff matrix is calibrated through experimental measurements taken from our testbed. In chapter 3, we identify a new (active) jamming attack/selfish behavior exploiting the carrier sensing functionality of 802.11 networks. We design, implement and evaluate a detection scheme, namely CMD (Carrier Misbehavior Detection) which achieves low false

positive and false negative rates. In chapter 4, we identify a second intelligent jamming attack at WLANs. In particular, a malicious node, can use extremely low power to affect only one client of the access point (AP). Consequently, due to the performance anomaly of 802.11 networks, the rest of the clients will be significantly affected. In order to detect and mitigate this implicit jamming attack, we design and implement FIJI. FIJI is a cross layer system that effectively detects implicit jamming attacks and subsequently shapes accordingly the traffic towards the clients with the goal of restoring benign operations. Our lightweight localization scheme is presented in chapter 5. The Packet Delivery Ratio is being minimized at the proximity of the jamming device. In order to search for these areas, we propose a localization algorithm based on the gradient descenet minimization method. We implement and experiment with our system on our indoor/outdoor wireless testbed. Our experimental validations show that a satisfactory performance is achieved by our localization strategy; we also identify and discuss scenarios where our scheme has a difficulty in localizing the jammer. After detecting the attack and/or localizing the jamming device, alleviating the jamming effects is needed. We take a less conservative approach as compared to frequency hopping and we *fight* against the jammer rather than simply avoid the latter. Chapter 6 describes the design, implementation and evaluation of our anti-jamming reinforcement system (ARES). ARES utilizies rate and power control to alleviate the transient and constant jamming effects. Our thorough evaluation on our indoor/outdoor testbed reveals the applicability of ARES in a large set of scenarios.

Chapter 2

On the Efficacy of Frequency Hopping in Coping with Jamming Attacks in 802.11 Networks

Frequency hopping (FH) has been the most popularly considered approach for alleviating the effects of jamming attacks. In this chapter, we re-examine the efficacy of FH with an extensive study based on both experimentation and analysis. *Our conclusions are that FH techniques are largely inadequate in protecting the network against jamming attacks.* In a nutshell, the problems with FH are: **(a)** the energy spill over between adjacent channels that are considered to be orthogonal, and **(b)** the small number of available orthogonal bands. In this chapter, we provide a novel, measurement-driven, game theoretic framework that captures the interactions between a communication link and an adversarial jammer, possibly with multiple jamming devices, in a wireless network employing FH. Our model accounts for both of the above limiting factors and is used to provide bounds on the performance of proactive frequency hopping in alleviating the impact of a jammer. The main contributions of our work are: **(a)** Construction of a measurement driven game theoretic framework which models the interactions between a jammer and a communication link that employ FH. **(b)** Extensive experimentation on our indoor testbed in order to quantify the impact of a jammer on

802.11a/g/n networks. Interestingly, we find that 802.11n devices can be more vulnerable to jamming attacks as compared with 802.11 legacy devices. We carefully analyze the reasons behind this observation. **(c)** Application of our framework to quantify the efficacy of proactive FH and validation of our analytical bounds across a variety of 802.11 network configurations. **(d)** Formal derivation of the optimal strategies for both the link and the jammer in 802.11 networks. Both our analytical and experimental results demonstrate that frequency hopping is largely inadequate in coping with jamming attacks in current 802.11 networks.

2.1 Introduction

The availability of commercial jamming devices makes it easy for malicious attackers to disrupt operations of a wireless network [7] [3]. Numerous jamming attacks have been reported in the recent past [14] [15] [16]; this makes the defense against such attacks very critical. A jammer continually emits electromagnetic signals on the medium in order to prevent legitimate data exchanges. In particular a jammer achieves its goal in a CSMA/CA network (e.g. 802.11, sensor networks) by exploiting two transceiver functionalities: **(a)** the MAC protocol requires a transmitter to sense the medium to be idle prior to transmitting its packet; thus, in the presence of illegitimate jamming packets on the medium, a node will defer its transmissions, and **(b)** the packets from the jammer collide with legitimate packets at the receiver. Both of the above effects cause a drastic degradation in the achieved throughput.

Traditionally, frequency hopping has been considered to be a solution that can help alleviate the effects of jamming; both proactive and reactive frequency hopping strategies have been proposed in the literature [1] [2] [17] [8]. The ease of implementation has made proactive frequency hopping more popular; reactive frequency hopping has associated synchronization challenges between the transmitter and the receiver (to be discussed). In this chapter, we construct a measurement-driven, analytical framework for quantifying the efficacy of proactive frequency hopping[1]. Our framework

[1]We consider proactive frequency hopping since a practically viable reactive strategy is yet to emerge.

accounts for two factors that affect such a strategy. First, the number of available orthogonal channels dictates the effectiveness of frequency hopping. Second, depending on the separation between adjacent orthogonal channels on the available spectrum, there might be an energy spill over between the bands. All prior efforts on frequency hopping assume that operating on a channel[2] that is orthogonal to that being used by a jammer - i.e., there is no overlap associated with the spectral masks - automatically protects a link. However if the aforementioned separation between bands is small, then a jammer (on a specific channel) can significantly hurt a legitimate communication that is on an adjacent orthogonal channel.

Our objective in this work is to understand the interactions between a jammer and a communication link and to quantify the efficacy of frequency hopping in coping with jamming attacks. In a nutshell, our contributions in this chapter are as follows:

1. ***Construction of a measurement-based game theoretic framework to capture the interactions between a link and a jammer employing proactive FH :*** We model the interactions between a legitimate link and the jammer as a two-player, zero-sum game. The strategies followed by each player and the payoff matrix account for the factors mentioned above. Our framework assumes that the jammer and the network, iteratively and selfishly try to adapt their strategies to stimulate the *best response* to the strategy of the opponent. Thus, the framework yields bounds on the performance of proactive frequency hopping. We extend our framework to cases with more than one jammer.

2. ***Quantifying the impact of a jammer via experiments on an indoor wireless testbed with both legacy 802.11 (802.11a and 802.11g) as well as its current 4G extension, 802.11n:*** We perform extensive experiments on our 802.11 indoor testbed in order to quantify the impact of a jammer that resides on channels that are orthogonal to the one used by a pair of legitimate transceivers. The results of our experiments show that the presence of a jammer on an adjacent, albeit orthogonal channel to that of the legitimate pair, can still degrade the performance of legacy 802.11 significantly. The throughput achieved by the legitimate pair can be reduced to just 10% of the throughput possible under benign conditions. This effect significantly limits the effectiveness of frequency

[2]We use the terms band and channel interchangeably.

hopping in 802.11 networks.

In addition, our experiments with 802.11n reveal additional vulnerabilities. 802.11n utilizes channel bonding as a way to increase the transmission rate [18]. In a nutshell with channel bonding, two or more adjacent channels are used in conjunction to form a new *wider* channel. Our measurements indicate that this property (in conjunction with the CSMA/CA policy inherited from legacy 802.11) can make 802.11n links more susceptible to jamming attacks. We provide a detailed discussion on why this is the case.

3. ***Applying our framework to quantify the efficacy of proactive frequency hopping in 802.11 networks:*** The measurements from our indoor testbed are then used to drive our framework, applying which we obtain bounds on the anti-jamming performance of a frequency hopping scheme in 802.11 networks. Our result indicate that proactive frequency hopping provides very limited protection to an 802.11 network, from jamming attacks. Our results show that with just 5 jammers one can basically block all the possible channels with 802.11a; this result is in stark contrast with previous efforts as per which, as many as 12 jammers are required to produce this effect.

4. ***Formal derivation of the optimal strategies for both the link and the jammer in 802.11 networks:*** We formally prove that the jammer has a *unique* optimal FH strategy when only a single jamming device is being employed. We extend the result for cases where multiple devices are used. We also prove certain key properties that have to be fulfilled by an optimal FH strategy, followed by a communication link.

Scope of our work: The main application of our framework is the evaluation of FH as a jamming countermeasure. We wish to point out however that our model captures the interactions between communication links and jammers when FH is used by all entities in the wireless network. As such, it can be used from both perspectives (the communication link's and the jammer's) and provide useful insights based on each player's objective.

The rest of the chapter is organized as follows. In section 6.2 we discuss related work in brief. Section 2.3 describes our measurement-driven, game theoretic framework. We describe our wireless testbed and the experimental methodology in section 6.3. In section 2.5, we present the ex-

perimental results that serve as measurement-inputs for our framework for an 802.11a/g network. Section 2.6 describes the application of our framework and the computation of performance bounds of a generic, proactive, frequency hopping scheme for the case of 802.11 networks; the optimal strategies are derived for both the legitimate communication pair and the jammer. We further validate our analytical results on our testbed. The performance of an 802.11n MIMO link under the presence of a jammer is considered in section 6.6. Section 6.7 discusses the applicability of our framework across a variety of jamming models, while our conclusions form section 6.8.

2.2 Background and Related Work

In this section we provide a brief overview on previously proposed frequency hopping schemes; we also discuss the practical limitations of these strategies.

2.2.1 Frequency Hopping Strategies

Frequency hopping strategies can be divided into two main categories.

Proactive frequency hopping: In a proactive frequency hopping scheme the pair of transceivers that form a link switch channels once every k seconds, irrespective of whether or not there is a jammer on the current channel. Gummadi *et al* [17] propose a rapid proactive frequency hopping scheme to alleviate the impact of specific patterns of narrow-band interference. Navda *et al* [1] implement a proactive frequency hopping protocol with pseudo-random channel switching for coping with a jammer. They compute the optimal residence time on a channel, assuming that the jammer is aware of the hopping protocol. However, they do not account for the energy spill over between adjacent orthogonal channels. A proactive strategy has the advantage of obviating the need for a jamming detection module. We wish to point out here that depending on the implementation, hopping between channels can also potentially incur a performance penalty due to the loss of throughput during the periods used for switching between frequencies [19]; however, in

professional implementations these penalties are likely to be extremely small.

Reactive frequency hopping: In a reactive frequency hopping scheme, a node switches to a new channel only if and when it detects the presence of a jammer. With such a scheme, when one member of a communicating node pair switches to a new channel, the other member will have to somehow detect the event and change its band as well. Hu *et al* [2] [8] propose a reactive channel hopping strategy. The key idea is that when a node is jammed it switches to a new but predetermined channel. The other node of the communicating pair switches to the same channel upon not hearing from its partner for a prolonged period of time. The authors point out the challenges in the implementation of such a strategy but do not provide solutions. In particular, there are issues related to synchronization, scalability, loss of packets and latency.

Given the ease of implementation, proactive frequency hopping strategies have been more popularly considered for coping with jamming. An effective reactive frequency hopping strategy is yet to emerge. Given this, we primarily consider a proactive approach in this work.

2.2.2 Practical Limitations of Frequency Hopping

Channel surfing (switching between channels) tries to avoid the jammer by switching between multiple orthogonal narrow spectral bands. The method can be effective in the presence of a narrow band jammer. In the presence of a wide band jammer that can simultaneously jam multiple bands (and in the extreme case, all possible bands) frequency hopping will not offer any benefits [4]. Given this, we only examine frequency hopping from the perspective of its effectiveness in coping with narrow band jammers.

The performance of frequency hopping will be limited by the extent to which an interferer on an adjacent (considered orthogonal) channel affects a considered channel [20] [21]. In [2] the authors take it for granted that 802.11a supports 12 *perfectly* orthogonal channels; this would imply that the presence of a jammer on one specific channel does not affect the other channels. In [17] the authors measure the throughput that is achieved when there is an interferer on a frequency band that

is $15MHz$ apart from the one being used by a legitimate communication. Given that the channel bandwidth with 802.11a is $20MHz$ ($22MHz$ with 802.11g), this scenario reflects the case of partially overlapped channels. The authors show that under these conditions, the overall throughput reduces to $2-3$ Mbps from the base rate of 6 Mbps; they conclude that 50% of the interference-free throughput is achievable if the interferer is present on a partially overlapped channel. We observe that the presence of a jammer on even *an adjacent orthogonal* channel ($20MHz$ apart from the channel of the legitimate communication) causes the throughput to drop to $3-4Mbps$. This is discussed in detail with our 802.11 measurements in section 2.5. We observe that the jamming-free throughput that is achievable on these links is around 27 Mbps (the links inherently support data rates that are much higher than the 6Mbps considered in [17]) and thus, the jammer degrades the throughput to about just $10-15\%$ of what is achievable. In summary, the presence of a jammer on an adjacent orthogonal channel can significantly hurt the performance of a legitimate communication; this in turn limits the effectiveness of frequency hopping strategies.

2.2.3 Game theoretic formulations of attacks

In the literature, game theoretic approaches have been used to model various wireless network problems. The work in [22] studies the problem of a legitimate node and a jammer transmitting to a common receiver and models it as a dynamic game. However, this work is theoretical; it suggests that the player that transmits with the highest power is the winner of the game. In contrast, our work is measurement driven and is validated via experimentation; it provides a comprehensive look at the performance of proactive frequency hopping in coping with jamming attacks. In [23], the authors examine the interactions between a single channel sensor network and a jammer. They are concerned with the detection of the jammer and more specifically, they try to minimize the detection time. They formulate and solve non-linear optimization problems to compute best responses of the attacker and the network to the worst-case strategy of the other. The authors of [24] use linear programming to model a specific class of attacks on network flows. Their work however, differs substantially from ours; it is not based on experimentation and does not consider channel surfing.

Liu *et al* [25] propose a novel approach SPREAD, to address the problem of cross layer DoS attacks in wireless data networks. They use a game theoretic approach to describe the interactions between a smart jammer that takes into account protocol specific parameters and the possible decisions of SPREAD. However, their work is neither based on experimentation nor does it examine the performance of frequency hopping.

Finally, in some more recent efforts, emulation attacks in cognitive communication systems are being cast as game theoretic problems. In particular, Li *et al* [26] study a primary user emulation jamming attack in a cognitive radio network utilizing game theoretic notions. The authors provide numerical solutions for different variations of the attack model and show that the performance of a secondary user is improved when the number of available channels is increased. Thomas *et al* [27] model the interactions between a selfish radio and a well behaved radio, as well as between two selfish radios, using the Bayesian game framework. They show that both types of interactions result in games with imperfect knowledge which can lead to Bayesian Nash Equilibrium (BNE) with both pure and mixed strategies. The also show that under different system parameters different BNEs arise.

2.2.4 Prior work on energy spill over between 802.11 channels

The authors in [28] try to exploit partially overlapped channels to improve the end-to-end application throughput. The efforts in [29] [30] and [31] try to understand the impact of the use of adjacent channels on a multi-radio, multi-hop 802.11 mesh network. Their findings indicate that multi-hop performance in mesh networks is affected by the adjacent channel interference that one NIC (Network Interface Card) imposes on the other NIC of the same node. However, none of the above efforts consider the presence of a malicious node, which injects packets on the medium to launch an attack.

To the best of our knowledge, our work is the first attempt to construct a measurement based analytical framework which quantifies the performance of a generic proactive frequency hopping strategy in coping with jamming attacks in any given wireless network.

2.3 Our Framework: The Generic Model of the Game

In this section we present our game which models the interactions between the legitimate communication link and the jammer. Both entities employ frequency hopping in order to achieve their objectives. On the one hand the link switches between bands in order to avoid the jammer; on the other hand the jammer hops across bands in order to find the communication link and hurt its performance. We model this interaction as a game. A game in normal form can be represented by a triplet $< N, (\Sigma_i), A >$. In this representation, N is the finite set of players, Σ_i is the set of possible strategies for player i and A is the payoff matrix of the game.

In our case the set N contains only two players; the jammer and the legitimate link. Both these players have the same set of strategies; $\Sigma = \{set\ of\ available\ orthogonal\ bands\}$. The payoff matrix should represent the objectives of each player. In our case the objective of the legitimate link is to increase its throughput by hopping channels - i.e. changing its strategy - while the objective for the jammer is to reduce this throughput. As a result, an appropriate definition for the payoff matrix is the following: $A_{i,j}$ is the percentage of the jamming-free throughput that the legitimate link enjoys when it resides on channel i with the jammer residing on channel j. With this definition of the payoff matrix, the value (or the payoff) v of the game is defined to be the percentage of the jamming-free throughput that is achieved on the link. On the one hand, the link is trying to maximize its payoff; on the other hand the jammer is trying to minimize the same payoff. As a result our game is a zero-sum, two person game. This means that **an equilibrium always exists** [32][3]. Our analysis yields the probabilities with which the legitimate link and the jammer ought to occupy the various channels in order to achieve the equilibrium performance.

The link chooses its channel randomly, using a probability distribution (mixed strategy) x, while the jammer picks its channel as per a probability distribution y. With this, the expected throughput achieved on the link (value of the game) is simply $v = x^T Ay$. We can always find the equilibrium strategies x^* and y^*, by solving the above game. The optimal mixed strategy x for the maximizing

[3]We wish to stress that our goal is not to provide a system that will compute this equilibrium in real time, but to quantify the performance of a proactive frequency hopping scheme.

player (the legitimate link) can be found by solving the following linear program:

$$\begin{aligned} \text{maximize} \quad & v && (2.1)\\ \text{subject to} \quad & A^T x \geq v && (2.2)\\ & |x| = 1 && (2.3)\\ & x \geq 0 && (2.4) \end{aligned}$$

and the optimal strategy y for the minimizing player (the jammer) is found as the solution to the dual linear program:

$$\begin{aligned} \text{minimize} \quad & v && (2.5)\\ \text{subject to} \quad & Ay \leq v && (2.6)\\ & |y| = 1 && (2.7)\\ & y \geq 0 && (2.8) \end{aligned}$$

In the above formulation, each of the constraints, (2.2) and (2.6), are used to describe the $|\Sigma|$ inequalities in a compact way. In particular, $A^T x$ and Ay are $|\Sigma| \times 1$ vectors, and each element of these vectors should satisfy the corresponding inequality with respect to v. Furthermore, $|x|$ is the 1-norm of vector x, i.e., the sum of all its coordinates. If both players play the game according to their equilibrium mixed strategies x^* and y^*, (computed by solving the above linear programs) the game would be in an equilibrium state. *At equilibrium, no player would benefit from changing the probability distribution with which they choose their channels.*

From the above formulation one can see that our framework accounts for both (i) the number of available orthogonal channels of the wireless technology under consideration and (ii) the effectiveness of a jammer which resides in a different orthogonal band. In the following sections we will show how we can apply our framework to an 802.11 network[4].

[4]We will also show how we can easily extend our framework to account for cases with more than one jammer.

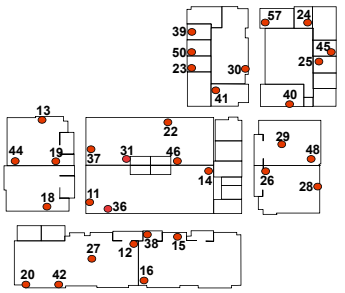


Figure 2.1: Deployment of our wireless testbed.

Note here that a probabilistic analysis could be used to model the interactions between a jammer and the communication link. However, as the dimensionality of the problem increases and/or the components of Σ change (e.g., different frequency allocations across large wireless networks), such an analysis is likely to increase in compexity or become intractable. Our game theoretic model on the other hand, is easily applicable in such contexts.

2.4 Experimental Setup

Prior to applying our framework to various 802.11 configurations, we describe our wireless testbed and the methodology followed in our experiments.

2.4.1 Testbed Description

Our 802.11a/g wireless testbed consists of 32 Soekris net4826 nodes [33]. Each node mounts a Debian Linux distribution with kernel v2.6.16.19 over NFS. The nodes are synchronized with an NTP server. The Soekris boxes have 2 miniPCI slots. These nodes are equipped with two miniPCI 802.11a/g WiFi cards; in particular, they have an *EMP-8602 6G* with the Atheros chipset and an *Intel-2915*. The layout of our testbed is depicted in Figure 2.1.

With our *EMP-8602 6G* cards, we use the MadWifi driver [34]. In addition, we use a proprietary version of the *ipw2200* AP and client driver/firmware with the *Intel-2915* cards. With this version we are able to tune the CCA (**C**lear **C**hannel **A**ssessment) threshold parameter; note that

this functionality has been implemented in the prototype firmware. The ability to tune the CCA threshold helps us implement a jammer as discussed later in this section.

The architecture of our 802.11n testbed is similar to the one described above. However, the nodes are utilizing 15 Soekris net5501 boxes[5], which are equipped with an *RT2860 mini-PCI* card that supports 802.11n communications.

2.4.2 Experimental Methodology

Our measurements are on a large set of individual links on our testbed. We perform experiments by varying the transmission powers of both the jammer(s) and the legitimate transceivers. We perform experiments with all modes, namely, 802.11a/g/n. Our experiments with 802.11g/n are conducted late at night in order to avoid interference from other co-located WLANs that operate at the same frequency band (note that RT2860 operate only in the 2.4GHz band in 802.11n mode). In our experiments, we have used all the orthogonal channels that are available with all modes of operation. There are only 3 orthogonal channels in the $2.4GHz$ band (i.e., 802.11g), while there are 12 orthogonal channels in the $5GHz$ band (i.e., 802.11a).

2.4.3 Implementing a Jammer

To facilitate our experiments, we implement our own jamming utility. The implementation of a jammer with an 802.11 legacy device has to ensure that: **(a)** other packets on the medium do not prevent the jammer from transmitting its packets, and **(b)** when active, the jammer should be able to send its malicious packets at the maximum possible sending rate in order to cause high impact on legitimate connections. The former requires the tuning of the CCA threshold, while the latter calls for the use of specific types of packets.

We implement our jammer on an 802.11 legacy device by setting the CCA threshold to a very high value ($\approx$ 0 dBm). This ensures that the device ignores the traffic in transit over the wireless

[5]These boxes have higher processing capabilities - as compared to net4826 -and can realize the MIMO benefits in terms of achievable throughput [35].

medium. We observe that packets always arrive at the jammer's circuitry with power less than 0 dBm even if the distances between the jammer and the legitimate transceivers are very small.

In order to ensure that the jammer continuously transmits packets on the medium, we have developed a user-space software utility. With this, the jammer continuously *broadcasts* UDP packets. Given that the backoff functionality is by default disabled in 802.11 for broadcast traffic, our software utility can ensure that packets are sent as fast as possible. With such transmissions the jammer does not wait for any ACK packets[6]. Our utility employs *raw sockets*, which allow the construction of a UDP packet from scratch and the forwarding of the packet directly down to the hardware, for transmission. Note here that such an operation requires administrative privileges. To summarize, our jammer utility consists of a specific NIC configuration that sets CCA=0 and a software utility for continuously generating and transmitting broadcast packets. The former feature is possible with our *Intel-2915* cards, since we have access to the firmware.

For our experiments we also utilized the *iperf* measurement tool to generate data traffic with packets of size 1500 bytes, on a legitimate link. Note that, we use the terms *the communication link*, *the link* and *legitimate link* interchangeably. We initiate traffic between the nodes and immediately after, we turn on the jammer(s). In the following section we present the results of our experiments.

2.5 Measuring the Impact of a Jammer in Legacy 802.11 Networks

In this section we present the measurements that will drive the payoff matrix of our game in the context of 802.11 networks. The measurements quantify the impact of a jammer that resides on a channel that is orthogonal to that of the communication link; we observe how this affects the performance of the legitimate link and incorporate these observations into our framework. We describe our experiments with both 802.11a and 802.11g.

[6]This configuration allows the deferral of back-to-back transmissions for the minimum possible time (i.e., $DIFS + min_{BackOff}$).

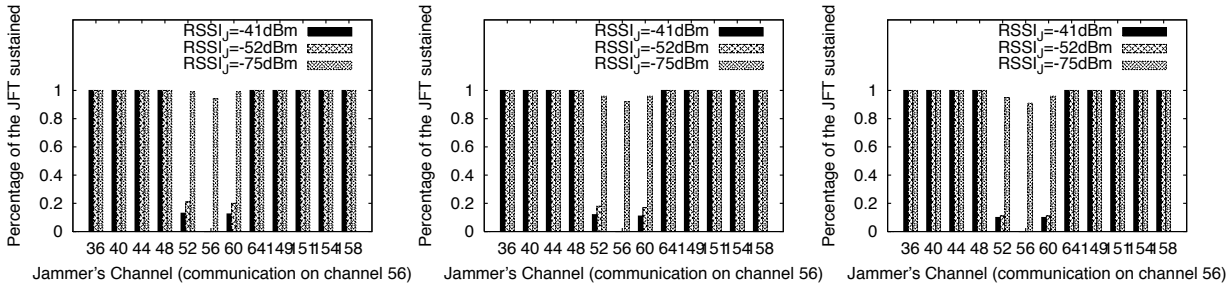


Figure 2.2: Percentage of the jamming free throughput (JFT) achieved when the jammer is on various channels, and for various $RSSI_J$, for the case of 802.11a. In the three figures we have $RSSI_l = -37dBm$, $RSSI_l = -47dBm$ and $RSSI_l = -66dBm$, respectively.

We use $\mathbf{RSSI_J} = \mathbf{max}(\mathbf{RSSI_{JT}}, \mathbf{RSSI_{JR}})$ to denote the maximum RSSI (**R**eceived **S**ignal **S**trength **I**ndicator) value that is observed on a link with regards to the signal from the jammer[7]. $RSSI_{JT}$ is the RSSI due to the signal from the jammer at the transmitter, while $RSSI_{JR}$ is the corresponding RSSI as observed at the receiver. As mentioned earlier, the jammer can affect both the transmitting and receiving functions of a node; in particular, it can cause interference at the receiver while it can cause the transmitter to defer its transmissions. By choosing the maximum value, we capture the case wherein the jammer has the maximum impact on the considered link. $\mathbf{RSSI_l} = \mathbf{min}(\mathbf{RSSI_{TR}}, \mathbf{RSSI_{RT}})$ denotes the minimum RSSI value between the end points of the communication link. $RSSI_{TR}$ is the RSSI of the signal from the transmitter at the receiver, while $RSSI_{RT}$ is the RSSI in the reverse direction. $RSSI_l$ represents the worst case RSSI for the link in the realistic scenario where the link is not symmetric.

2.5.1 Impact of Jamming in 802.11a

The 802.11a standard supports 12 orthogonal bands or channels. Each of these channels is of $20MHz$ bandwidth. The spacing between the central frequencies of these bands is $20MHz$ as well. In general, when two links communicate on orthogonal bands it is assumed that one does not interfere with the other. This observation drives all the frequency hopping schemes proposed thus

[7]This is measured when both the jammer and the communication link are on the same channel.

far. These schemes assume that via a transition to a channel that is orthogonal to that of the jammer, a communication link can be completely protected. However, this assumption does not hold with two adjacent orthogonal channels. We first present our experimental results to demonstrate this and later, discuss the reasons for this effect.

In our experiments a legitimate connection is initiated on one of the 12 orthogonal channels of 802.11a. Subsequently, the jammer is turned on. The jammer sequentially sweeps the 12 orthogonal channels, one channel at the time. We measure the throughput of our legitimate connection in each case. We repeat the experiments for various $RSSI_J$ and $RSSI_L$ values, in order to account for various topologies. In Figure 2.2 we present the results for the case where the communication channel was channel 56. The results were similar when the legitimate connection was established on any other different channel.

Our main observation is that ***a jammer which transmits signals on an orthogonal band that is adjacent to that of the legitimate communication, can significantly degrade the throughput performance. Specifically, the throughput of the connection drops to approximately 10 to 15 % of the jamming-free throughput.*** The exact degradation depends on the distance between the jammer and the link and the corresponding channel characteristics. However, our measurements indicate that when $RSSI_J \gg CCA$ for a co-channel user, that user gets at most 15% of the jamming-free throughput if it were to use the adjacent orthogonal bands. The reason for this may be attributed to the fact that RF filters typically do not provide sharp cut-offs at the specified boundaries of the channels [36]. As a result, the spectral power from the signal in one channel (that of the jammer) may spill over to an adjacent channel (that of the legitimate communication), even if in theory they are considered orthogonal. In order to completely avoid the effects of jamming, the legitimate connection will have to be at least 2 orthogonal channels apart from the channel on which the jammer is present.

Next, we conducted experiments with two jammers. We considered all possible placements of the jammers on the 12 orthogonal channels. Our main observations are summarized in figure 2.3. When the two jammers reside on the two orthogonal channels adjacent to that of the communication

link, the degradation in the link throughput can be as high as 95%.

We would like here to emphasize the fact that the above observations do not hold for channels 64 and 149. These channels are more than $400\ MHz$ apart and as our measurements indicate are completely isolated.

We use these measurements as inputs to our game-theoretic framework in section 2.6.

2.5.2 Impact of a Jammer With 802.11g

In contrast with 802.11a, 802.11g has only 3 orthogonal channels, each of which is of $22MHz$ bandwidth. The central frequencies of these bands are however, $25MHz$ apart. This implies that there is a *secure zone* of $3MHz$ between the adjacent orthogonal channels. Conducting the same experiments as before, we obtain the results in Figure 2.4.

As with 802.11a, we observe that in the presence of a jammer on an orthogonal, adjacent channel, the performance of a legitimate connection is still degraded. However, with 802.11g the degradation is significantly lower. This can be primarily attributed to the *larger* channel separation between adjacent orthogonal channels; this results in a reduced seepage of the spectral power of the jammer into the adjacent channel being used by the legitimate connection. However, since there are only 3 orthogonal bands in 802.11g, frequency hopping is not expected to be very effective.

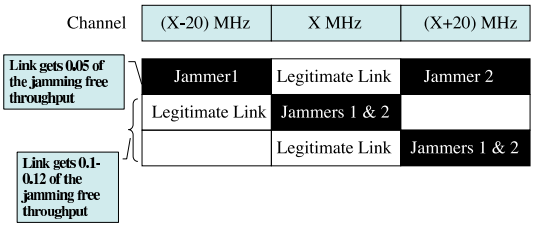


Figure 2.3: The case of 2 jamming nodes on adjacent communication channels.

2.6 Applying and Validating our Framework in Legacy 802.11 Networks

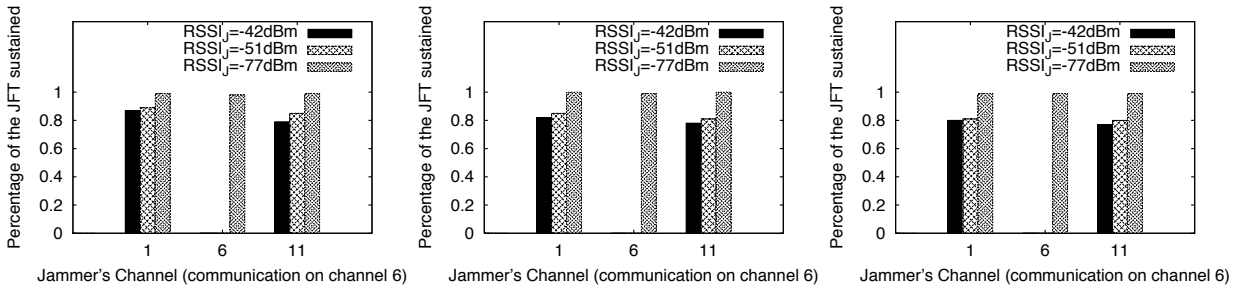


Figure 2.4: Percentage of the jamming free throughput (JFT) achieved when the jammer is on various channels, and for various $RSSI_J$, for the case of 802.11g. In the three figures we have $RSSI_l = -39dBm$, $RSSI_l = -45dBm$ and $RSSI_l = -68dBm$, respectively.

In this section we will apply our game-theoretic framework based on the measurements presented in the previous section.

2.6.1 Model for 802.11a

An 802.11a wireless network can support twelve orthogonal channels[8]. For ease of presentation, we label the channels: 1, 2, . . ., 12. The central frequencies of the channels are 20MHz apart, with the exception of the eighth and ninth channel pair that are 425MHz apart. Based on the measurement results obtained in the previous section, if the jammer is on a channel that is adjacent to that of the link (with the exception of the eight and ninth channel pair), we assume that the link can achieve only 12% of its jamming-free throughput; if the jammer is on the same channel as that of the link, no throughput is achieved. If two jamming devices reside on the two adjacent channels of the link, the throughput achieved on the link is just 5% of the jamming-free throughput. Again, the eighth and ninth channels are very far apart and so, if the link resides on one of those channels and the jammer is on the other one, then the link 's performance is not deteriorated. Note here that, if the

[8]802.11a channels are 36, 40, 44, 48, 52, 56, 60, 64, 149, 153, 157, 161 in North America.

link were to operate on any of the channels 1, 8, 9 or 12, the jammer would only impact the link if it resides on the same channel or the immediate adjacent channel; for the other cases, there are two such possible adjacent channels.

First, we consider the case where the communication link is on channel i and we have a single jamming device on channel j. The payoff matrix is then given by:

$$A^{1,a}_{i,j} = \begin{cases} 0 & \text{if } i = j, \\ 1 & \text{elseif } (i = 8 \text{ and } j = 9) \text{ or } (i = 9 \text{ and } j = 8), \\ 0.12 & \text{elseif } |i - j| = 1, \\ 1 & \text{otherwise.} \end{cases}$$

We can now use the linear programs (2.1)-(2.4) and (2.5)-(2.8) in order to compute equilibrium strategies for the link and the jammer respectively. First, let us consider the scenario where there is just one jamming device. Then, the mixed strategies x^* and y^* are shown in Tables 2.1 and 2.2.

channel j	1	2	3	4	5	6	7	8	9
y^*_j	.0894	.1155	0	.1016	.1016	0	.1155	.0894	.1728

channel j	10	11	12
y^*_j	.0207	.0207	.1728

Table 2.1: Mixed strategy for one jamming device in 802.11a

channel i	1	2	3	4	5	6	7	8	9
x^*_i	.1910	0	.026	.0894	.0894	.0260	0	.191	.1728

channel i	10	11	12
x^*_i	.0207	.0207	.1728

Table 2.2: Mixed strategy for the communication link in 802.11a

The strategy y^* gives the probability distribution as per which the jammer should choose the next channel to hop. We show that the equilibrium strategy for the jammer is unique. For the link, x^* is one possible equilibrium probability distribution according to which the next channel can be chosen; however, it is not unique. If the players play as per these equilibrium strategies, the value of the game is $v = 0.809$. This implies that the expected throughput on the link is about 81% of its

jamming-free throughput.

Uniqueness: The following corollaries formally prove that (i) the jammer's equilibrium strategy is unique and, (ii) the link should not use channels 2 and 7.

Corollary 1. *The linear program (2.5)-(2.8), with $A = A^{1,a}$, has just one optimal solution $y = y^*$, where y^* is given in Table 2.1.*

Proof. We prove the corollary by contradiction. Let there be another optimal solution $\hat{y} \neq y^*$. In other words, if possible, let there be a solution $\hat{y}$ with a non-zero 1-norm distance from y^*. The 1-norm distance is defined as $|\hat{y} - y^*| = \sum_{i=1}^{12} |\hat{y}_i - y_i^*|$. If we cannot find such a solution $\hat{y}$, then the solution y^* is unique. In other words, we want to check if the following optimization problem has a zero objective value or not. The optimization problem that we want to solve is:

$$\begin{aligned} \text{maximize} \quad & |\hat{y} - y^*| && (2.9) \\ \text{subject to} \quad & A\hat{y} \leq 0.809 && (2.10) \\ & |\hat{y}| = 1 && (2.11) \\ & \hat{y} \geq 0 && (2.12) \end{aligned}$$

The above formulation is not a linear program (the objective function is non-linear). We reduce the problem into solving $2 \cdot 12 = 24$ linear programs below. For each of the linear programs, our goal is to check if the objective function is zero.

For $i = 1, \ldots, 12$,

$$\begin{aligned} \text{maximize} \quad & \hat{y}_i - y_i^* && (2.13) \\ \text{subject to} \quad & A\hat{y} \leq 0.809 && (2.14) \\ & |\hat{y}| = 1 && (2.15) \\ & \hat{y} \geq 0 && (2.16) \end{aligned}$$

$$\begin{aligned} \text{maximize} \quad & y_i^* - \hat{y}_i && (2.17)\\ \text{subject to} \quad & A\hat{y} \leq 0.809 && (2.18)\\ & |\hat{y}| = 1 && (2.19)\\ & \hat{y} \geq 0 && (2.20) \end{aligned}$$

By solving each of the above linear programs, we verify that the objective value is zero. This proves the uniqueness of solution y^*. □

Corollary 2. *Any equilibrium strategy x^* for the maximizing player (the link) has $x_2 = x_7 = 0$.*

Proof. To prove that in any optimal solution, $x_2 = x_7 = 0$, we formulate the following linear program.

$$\begin{aligned} \text{maximize} \quad & x_2 + x_7 && (2.21)\\ \text{subject to} \quad & A^T x \geq 0.809 && (2.22)\\ & |x| = 1 && (2.23)\\ & x \geq 0 && (2.24) \end{aligned}$$

The linear program tries to find the maximum value for the sum $x_2 + x_7$ under the constraint that the achieved payoff is at least 0.809 (this is the maximum achievable payoff). The solution to the above linear program yields an objective value of zero. In other words, there cannot be any optimal solution with either $x_2 \neq 0$ or $x_7 \neq 0$. □

Corollary 3. *If the jammer plays the strategy of corollary 1, then the link player can set $x_1 + x_3 + x_4 + x_5 + x_6 + x_8 + x_9 + x_{10} + x_{11} + x_{12}$ to any non-negative value, as long as their sum is 1.*

Proof. The value of the game is $x^T Ay$. Substituting $A = A^{1,a}$ and $y = y^*$ we have:
$v = x^T Ay = 0.809(x_1 + x_3 + x_4 + x_5 + x_6 + x_8 + x_9 + x_{10} + x_{11} + x_{12}) + 0.8059(x_2 + x_7)$
In order to maximize v we should set $x_2 = x_7 = 0$, and then set the remaining variables to any non-negative values such that $x_1 + x_3 + x_4 + x_5 + x_6 + x_8 + x_9 + x_{10} + x_{11} + x_{12} = 1$. □

Recall that the solution x^*, provides the best response strategy of the communication link to the strategy y^* of the jammer (and vice versa). The set of channels available can be separated into two disjoint sets in terms of interference, that is, channels 1-8 and 9-12. In the first subset, the jammer picks channels 2 and 7 with the highest probability, since it can then block a set of 3 channels that cannot be simultaneously blocked otherwise. As a result, the link should avoid these channels (i.e., $x_2 = x_7 = 0$) and place its device with high probability on the *edge* channels (i.e., 1, 8, 9 and 12). In the second subset, the jammer picks the edge channels with higher probability, since it then can effectively block channels 9-12. Note here that, if we were to compare the probabilities with which the edge channels are occupied by the link, we have $x_9 = x_{12} < x_1 = x_8$, because $y_9 = y_{12} > y_1 = y_8$.

Multiple jammers

We consider the scenario where the jammer can employ more than one jamming device, that is, it can block more than one channel. This case of multiple jammers can still be modeled as a zero-sum two-player game and described by a matrix A_{ij}. Here i is the channel on which the link resides and j represents the channels where the jamming devices reside. In order to reduce the dimension space due to the multiple jamming devices, we use a row/column major order representation. As an example, let us consider the case of two jamming devices on channels j_1 and j_2. There are $12^2 = 144$ possible placements of these devices on the frequency spectrum. Each placement can be encoded by a single value j. It is easy to see that by setting $j = 12(j_1 - 1) + j_2$, every combination of j_1 and j_2 is encoded into a unique value.

Table 2.3 summarizes the expected percentage of the jamming-free throughput in equilibrium for the case of one, two, three, four and five jamming devices.

# jammers	1	2	3	4	5
v	80.9%	61.8%	42.7%	23.6%	4.5%

Table 2.3: Expected link throughput for 802.11a, using different numbers of jammers

It is straightforward to extend Corollary 2 and Corollary 3 for the multiple jammer cases. Thus,

x^* given in Table 2.2 is an equilibrium strategy for any of the cases. The jammer's equilibrium strategy is no longer unique but still y_3 and y_6 are 0. Moreover, it makes no sense to put multiple jamming devices on the same channels.

Sensitivity to measurements: The results thus far, were based on a premise that if the link was on a channel that was adjacent to that being used by the jammer, only 12% of its jamming-free throughput can be achieved. Note that in practice, the exact degradation experienced varies depending on the locations of the link and the jammer and the environment. Our experiments suggest that only up to 10-15% of the jamming free throughput is achieved. Using any other value in this range for the payoff matrix would not change the results significantly (at most 3% change).

2.6.2 Model for 802.11g

The model for 802.11g is simpler to solve, given that there are just three orthogonal channels. For one jamming device the payoff matrix is:

$$A^{1,g}_{i,j} = \begin{cases} 0 & \text{if } i = j, \\ 0.88 & \text{if } |i-j| = 1, \\ 1 & \text{otherwise,} \end{cases}$$

For two jamming devices the payoff matrix is given by

$$A^{2,g}_{i,j_1 j_2} = \begin{cases} 0 & \text{if } i = j_1 \text{ or } i = j_2, \\ 0.88 & \text{elseif } |i-j_1| = 1 \text{ or } |i-j_2| = 1, \\ 1 & \text{otherwise,} \end{cases}$$

Note here that interestingly, our measurements indicate that for a link that is being (partially) jammed by a jammer residing on an adjacent channel, adding one more jammer on the other adjacent orthogonal channel does not further impact the link's throughput (as it does in the case of 802.11a). This can be attributed to the relatively large spectral zone with 802.11g; additional energy spillage is negligible. For three jamming devices, all values in the payoff matrix are zero:

$$A^{3,g}_{i,j_1 j_2 j_3} = 0$$

Again, solving the game using linear programming, we get the equilibrium strategies for both players and the expected payoffs (percentage of the link's jamming-free throughput). These payoffs are summarized in table 2.4.

# jammers	1	2	3
v	61.46%	29.33%	0%

Table 2.4: Expected link throughput for 802.11g, using different numbers of jammers

With one jamming device, both players have the same equilibrium strategy; the strategy is tabulated in table 2.5.

channel i	1	2	3
y^*_i	0.3492	0.3016	0.3492
x^*_i	0.3492	0.3016	0.3492

Table 2.5: Mixed strategy for the link and one jamming device in 802.11g

If the jammer has two jamming devices, they should be activated in pairs so as to maintain a uniform probability of using each channel. The communication link should also hop among the three channels, uniformly at random. The strategies are shown in tables 2.6 and 2.7.

channels (j_1, j_2)	(1,2)	(1,3)	(2,3)
$y^*_{j_1,j_2}$	0.3333	0.3333	0.3333

Table 2.6: Mixed strategy for the two jamming devices in 802.11g

With three or more jamming devices, no throughput can be achieved on the link with 802.11g, as one might expect. Next, we prove the uniqueness of the above solutions.

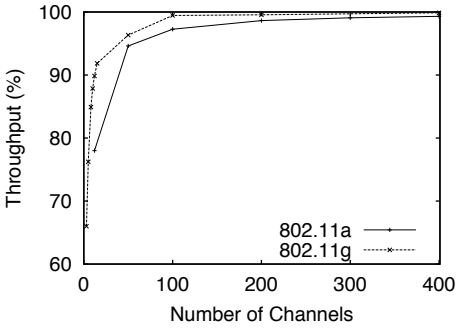


Figure 2.5: Increasing the spectrum availability, significantly increases FH's robustness against jamming.

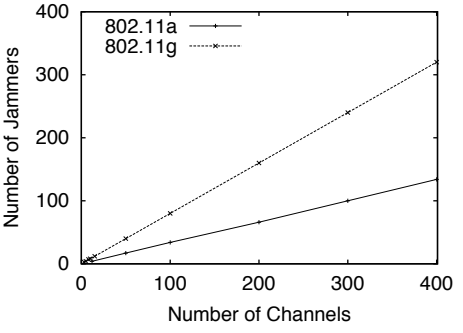


Figure 2.6: Number of jammers needed to drop throughput below 20% of the jamming free performance enjoyed.

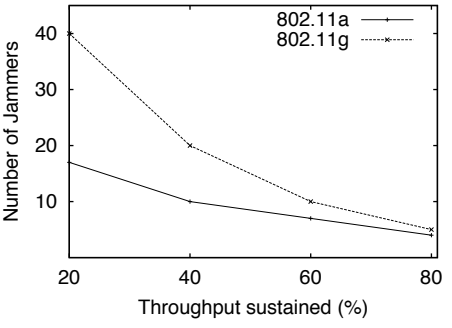


Figure 2.7: Number of jammers needed to drop the throughput at a specific percentage (50 channels).

channel i	1	2	3
x_i^*	0.3333	0.3333	0.3333

Table 2.7: Mixed strategy for the communication link against two jamming devices in 802.11g

Corollary 4. *The solution given in table 2.5 is the unique optimal solution for the linear programs (2.1)-(2.4) and (2.5)-(2.8), for* $A = A^{1,g}$.

Proof. We prove the corollary for the solution of the dual linear program (2.5)-(2.8); a similar proof can be easily constructed for the primal optimal solution x^* in table 2.5. An optimal solution

$y = y^*$ given by Table 2.5 makes all the constraints tight i.e.,

$$0.88y_2 + y_3 = v \tag{2.25}$$

$$0.88y_1 + 0.88y_3 = v \tag{2.26}$$

$$y_1 + 0.88y_3 = v \tag{2.27}$$

In order to prove this, consider the following:

a) some $\delta > 0$ is subtracted from y_1 and added to y_2 or y_3 or both. Then, the first constraint will yield a value more than v. **b)** some $\delta > 0$ is subtracted from y_2 and added to y_1 or y_3 or both. Then, the second constraint will yield a value more than v. **c)** some $\delta > 0$ is subtracted from y_3 and added to y_1 or y_2 or both. Then, the third constraint will result in a value more than v. **d)** some $\delta_1 > 0$ is subtracted from y_1, some $\delta_2 > 0$ is subtracted from y_2, and $\delta_1 + \delta_2$ added to y_3. Then, the first constraint will yield a value more than v. **e)** some $\delta_1 > 0$ is subtracted from y_2, some $\delta_2 > 0$ is subtracted from y_3, and $\delta_1 + \delta_2$ added to y_1. Then, the third constraint will have value more than v. **f)** some $\delta_1 > 0$ is subtracted from y_1, some $\delta_2 > 0$ is subtracted from y_3, and $\delta_1 + \delta_2$ added to y_2. Then, the sum of the first and the third constraints will be more than $2v$. With this, either the first or the third constraint must result in a value more than v. Thus, there is no way to construct another feasible solution with a value at most v. In other words, the solution in table 2.5 is unique. □

2.6.3 The Effect of Number of Channels

The number of available channels is a limiting factor on the applicability of frequency hopping in current commodity systems. In this section we want to quantify the efficiency of frequency hopping in coping with jamming with a varying number of orthogonal bands. In other words, we ask the question "what if the commodity systems had higher numbers of orthogonal bands?"; to what extent would it improve the effectiveness of frequency hopping in avoiding a jammer? We solve our game by calibrating a payoff matrix from our measurements but the matrix is appropriately expanded in order to emulate the existence of more channels. In particular, the effect of a jammer residing at an

orthogonal band is assumed to be the same as is in current commodity 802.11 systems. We find the solution to our two-player game with new payoff matrices derived from measurements with both 802.11a[9] and g. The results are presented in figure 2.5. We see that if a fairly large number of channels were available, then frequency hopping would be a very efficient anti-jamming technique. In particular, with a single jammer, the throughput is almost completely restored if the number of channels is close to 100.

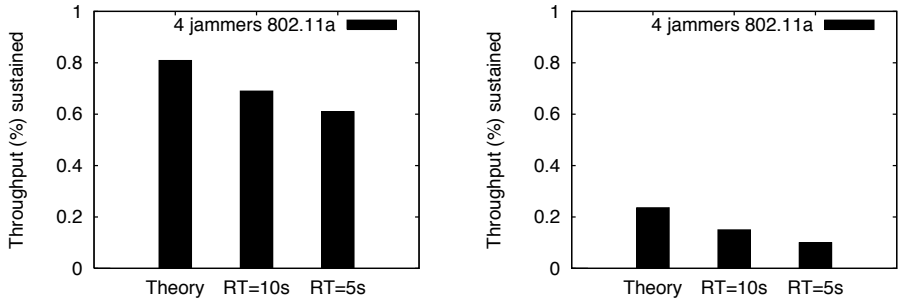
In figure 2.6 we present the number of jamming devices that one would need in order to bring the throughput down to below 20% of the jamming free performance. We notice that the number of devices needed for the model calibrated with measurements using 802.11g are higher than with the model based on 802.11a. This is due to the reduced effect that a jammer residing on an adjacent orthogonal channel has with 802.11g given that the channel spacing is larger. In particular, if 100 channels were available, with the energy spillage between orthogonal channels as with 802.11g, about 80 jammers would be necessary; in the corresponding case, with the energy spillage as with 802.11a, only about 34 jamming devices are sufficient.

Finally in figure 2.7 we present the number of jamming devices needed in order to drop the throughput of the link to a specific percentage of the jamming free throughput (x-axis) for a fixed number of channels (50). Again notice, that the jammers will be much more effective if the energy spillage between adjacent channels is higher (as with 802.11a).

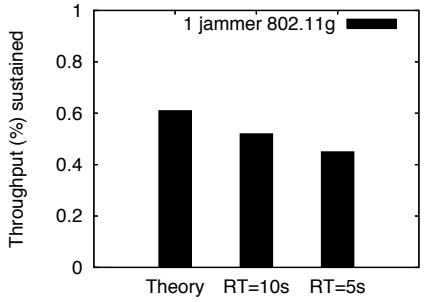
In summary, as one might expect, our results suggest that if current systems could support a larger number of orthogonal bands, frequency hopping has the potential of being a robust anti-jamming technique.

From a different point of view, we are interested in examining the effect of one or multiple jammers in a scenario where two adjacent orthogonal channels are completely isolated (i.e., $A_{i,j} = 0$). Such scenarios can exist if we were able to (i) reallocate the available bandwidth in such a way that adjacent orthogonal channels are isolated (which would result in fewer channels as compared to cur-

[9]For ease of presentation, here we assume that the central frequencies of all the channels are 20MHz apart. Although this might not be true (i.e. the cases of channels 64 and 149 with 802.11a that are 425MHz apart) it only affects the results by a negligible factor if the number of such pairs is small compared to the total number of channels.

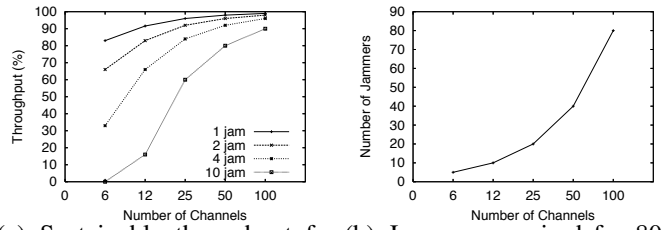


(a) 1 jammer with 802.11a. (b) 4 jammers with 802.11a.



(c) 1 jammer with 802.11g.

Figure 2.8: Experimenting with our prototype proactive FH. Our framework, indeed, bounds the performance of FH as jamming countermeasure.



(a) Sustainable throughput for different numbers of jammers. (b) Jammers required for 80% degredation.

Figure 2.9: Isolated orthogonal channels.

rent systems), or (ii) use additional resources/bandwidth and assure that the frequency bands used do not interfere with each other. These results, can provide useful guidelines for future frequency allocation policies that are resilient to jamming attacks.

Figure 2.9(a) depicts the sustainable throughput for different number of jamming devices versus the number of isolated frequency bands. As one might expect, increasing the number of isolated frequency bands, causes frequency hopping to be more robust to jamming attacks. As an example, with 100 isolated channels, even under the presence of 10 jammers the sustainable, jamming-free,

throughput is as high as 90%. Furthermore, it is interesting to notice, that if we were to reallocate/reassign the 5 GHz band in such a way that there is a 20MHz spacing between the channels (which results in 6 orthogonal bands), the sustained throughput with one jammer is 83%, with 2 jammers is 66% and with 4 jammers is 33%. All these values are higher than the corresponding values with the current 12 channel allocation (i.e., 80.9%, 61.8%, 23.6%).

Finally, figure 2.9(b) presents the number of jamming devices required for a sustained throughput of at most 20% as compared with the jamming free environment. As with current channel allocations, the number of jammers required increases as the number of available, isolated bands increases.

2.6.4 Validation Of Our Framework

In this section we build a proof of concept prototype of a proactive frequency hopping scheme. Note that our goal is to validate the performance bounds that were theoretically computed in the previous section and *not* the implementation of a *full fledged* distributed implementation of a frequency hopping technique.

System design and implementation

Our system implements a simple, generic proactive frequency hopping scheme. The scheme is based on the game described in the previous section. In particular, the network nodes switch between the available frequency bands, once every k seconds. The hopping sequence is known by all network nodes, but not by the jammer. This is achieved by an offline computation of the hopping sequence by using the linear programs from the previous section and a priori loading of the computed sequence on all the nodes in the network. A similar procedure is followed for the jammer's hopping sequence. An offline emulation of the sampled frequencies demonstrated that the system converges after approximately 70 frequency hops. Accordingly, we create various sequences of 100 frequencies each and experiment with them.

An important design parameter is the *residence time* of a node on the channel (denoted RT from now on). RT is defined to be the time that a node spends on a channel prior to hopping to a different channel. In the first set of experiments described in this section, we use *fixed* RT values of 5 and 10 seconds for both the jammer(s) and the link. Optimizing the RT is beyond the scope of this thesis. However, we experimentally study a plurality of scenarios where the jammer and the link use different RT values and discuss the implications thereof later in this section.

All nodes are synchronized using the Network Time Protocol (NTP) [37] through our testbed server. Thus, all nodes share the same clock and hop between the channels simultaneously. The hopping is implemented using the `ioctl()` [38] interface. The delay that ioctl() interface imposes is of the order of μsec [39] [40], and as a result the overall performance is not affected. The reader should also recall, that implementing a *professional*, proactive frequency hopping scheme is beyond the scope of this thesis, as mentioned in the beginning of this section.

Experiments with 802.11a

We perform experiments on several 802.11a links with jammers in their vicinities. Both the link being considered and the jammer, hop frequencies as per the equilibrium schedule (as discussed earlier). In particular, we conduct experiments with: **(a)** 40 different links on our testbed and, **(b)** 30 different equilibrium hopping sequences. Each of these hopping sequences consist of 100 sequential frequency hops for both the link under consideration and the associated jammer. The hopping sequences are samples generated with the probabilistic distributions from the output of our game theoretic framework, **(c)** 1, 2, 3 and 4 jammers active at a time, **(d)** $RT = 5sec$ and $RT = 10sec$. Note that in all our experiments we have used the Sample rate algorithm [41] (the default settings).

The results from our experiments with one active jammer are shown in figure 2.8(a). We observe that in practice, the throughput achieved in the presence of a jammer with a proactive frequency hopping strategy is lower than what is theoretically expected. This is because the model used in section 2.3 assumes *zero dwell times* between the channel hops, and perfect synchronization.

Neither of these assumptions holds in a real deployment. Furthermore, note that the throughput is lower due to a higher switching[10] and synchronization overhead if $RT = 5sec$ as compared to the case where $RT = 10sec$. In practice there is never perfect synchronization, even with NTP.

We experimented with 2, 3 and 4 jammers with similar results. In figure 2.8(b) we present the results for 4 jammers. We notice again that in practice the performance is poorer as compared to what is theoretically expected. In particular, with 4 jammers the throughput achieved is *only 8-10%* of the jamming free throughput.

Experiments with 802.11g

We report experiments with only one jammer with 802.11g. Our experiments suggest (as one might expect from our analysis) that the performance degrades significantly with 2 jammers and with 3 jammers the entire spectrum is blocked. As with 802.11a, we compute the equilibrium hopping sequences for both the link and the jammer, and experiment with two different values of RT. The hopping sequences were again of length 100. As previously, it was verified offline that 100 hops were enough for the game to converge to its optimal value. The results are shown in figure 2.8(c). As with 802.11a, we observe that the performance in practice is lower than what is theoretically expected (due to the same reasons as before).

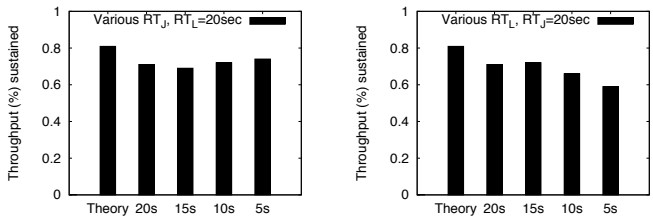
The sensitivity to the choice of RT

Our framework provides long term performance bounds and as a result, by itself does not yield insights on the right choice of the value of RT (for either the link or the jammer). Computing the optimal value for RT is beyond the scope of this thesis. However in our experiments we provide results when the link and the jammer have different values for this parameter (RT_L and RT_J, respectively).

In figures 2.10(a) and 2.10(b) we present the results of our experiments with 802.11a for the

[10]Note that with appropriate driver/firmware modifications - specific to the hardware in use - one can make this penalty extremely small.

case of a single jammer. First, in figure 2.10(a), we hold the RT for the link fixed at $20sec$. The RT of the jammer is varied. Reducing the RT value of the jammer can have two conflicting effects. On the one hand, the jammer can hit multiple channels during the $20sec$ RT period of the link; this can increase its effectiveness. On the other hand, it might incur a switching penalty each time it switches channels. We observe that when the RT of the jammer is reduced from $20sec$ to $15sec$, the first factor has a higher impact; however, further reducing the value of RT causes the second factor to be dominant. A similar behavior is observed when we keep $RT_J = 20sec$ and we vary RT_L. The sweet spot again appears when $RT_L = 15sec$.



(a) Varying RT_J. (b) Varying RT_L.

Figure 2.10: Validation of our framework.

We wish to point out that irrespective of the choice of RT, the practical schemes cannot do better than what is theoretically predicted by our framework in the long term. Our framework is independent of the RT of each player and the potential switching penalty (note that our analysis implicitly assumes zero switching penalty). Thus, although the performance of a frequency hopping strategy might be improved by tuning the frequency with which the link switches between channels, it is still limited and cannot provide better performance in the long run, than what is predicted by our framework.

2.7 Experimenting with 802.11n.

The use of antenna arrays or MIMO (multi-input multi-output) technology promises higher reliability; the 802.11n standard supports transmissions on MIMO links. In this section, our objective is to evaluate the efficacy of frequency hopping in 802.11n networks against jamming attacks.

A technique that is exploited to allow transmissions at higher rates with 802.11n is channel bonding. In a nutshell, we find that channel bonding makes frequency hopping less effective with regards to jamming attacks. We begin this section with a brief overview of channel bonding; subsequently we apply our game theoretic framework and evaluate the performance of 802.11n in the presence of a jamming attack.

2.7.1 Channel Bonding

802.11n devices can operate on channels that span either $20MHz$ *or* $40MHz$ bandwidth. In the latter case, channel bonding is used [18]. With channel bonding, two or more adjacent channels are used in conjunction to form a new *wider* channel. The expansion helps achieve higher data rates (practically doubles the possible rate). The thesis is that, the increased reliability possible on MIMO links (due to diversity and the use of space time codes) [42] can support transmissions at higher rates[11]. To elucidate the concept of channel bonding, consider channel 6 (as specified with 802.11g). Without channel bonding, the 802.11n signal utilizes the spectrum between $2.427MHz$ and $2.447MHz$. However, with channel bonding the spectrum that is used spans the frequencies between $2.417MHz$ and $2.457MHz$.

At this point we should note that 802.11n systems employ carrier sensing for medium access. This makes them susceptible to interference due to collocated links operating at the same or overlapping frequencies [35]. Consequently, 802.11n systems cannot take full advantage of the benefits of the underlying PHY layer technology (e.g. interference cancellation, support of simultaneous multiple transmissions etc).

2.7.2 MIMO Performance Under Jamming

As mentioned, MIMO links with Space-Time Block Codes (STBC) are expected to provide robustness to signal variations. Thus, the required SINR for achieving a target bit error rate is expected to

[11] With SISO, the higher the transmission rate, the lower the reliability.

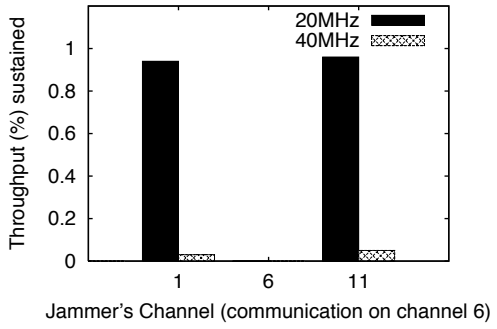


Figure 2.11: Channel bonding can degrade MIMO performance under jamming.

be lower than the corresponding requirement with SISO (Single-Input Single-Output) links[12].

For our experiments we use Ralink's RT2860 chipset, which supports 802.11n communications [43]. These cards operate in the $2.4GHz$ band. We used channels 1, 6 and 11 for our experiments; these are essentially, the only orthogonal channels in this band. We experimented with 40 MIMO STBC links on our testbed, each of which was under the influence of a jammer. Our experiments include both the cases of 20 and $40MHz$ bandwidth for the link, while the jammer uses a bandwidth of $22MHz$ (802.11g mode). Figure 2.11 depicts the results from our experiments. We only present the case where the communication is taking place on channel 6; other cases yielded very similar results.

From figure 2.11 we observe that the performance of 802.11n in the case where a bandwidth of $20MHz$ is used, is almost the same as that with 802.11g. 802.11n seems to offer a slightly better performance than 802.11g; an adjacent orthogonal jammer degrades the performance by only 5%. This can be due to two reasons: (a) MIMO links and STBC offer marginally better robustness to the jammer than SISO and (b) the secure zone with 802.11n is $5MHz$ as compared to $3MHz$ with 802.11g (with 802.11n the bandwidth is $20MHz$ while with 802.11g, it is $22MHz$).

The results with channel bonding show that the effectiveness of the jammer is dramatically increased in this case. The reason for this is that channel bonding practically eliminates orthogonality. Even if a jammer operates on a band of $22MHz$ (as in our experiments) and is active on the furthest

[12]Due to CSMA/CA though, such benefits might become obsolete if the transmitter can sense the jamming signals.

channels (i.e., channels 1 or 11) from that of the link (channel 6) there is an overlap; in other words, the jammer's signals interfere with the link. The link is safe only when it operates on channel 1 and the jammer occupies channel 11 and vice versa.

Based on the above measurements, we use the framework presented in section 2.3 to quantify the performance of a proactive frequency hopping strategy with 802.11n. Applying the model to the case where a bandwidth of $20MHz$ is used yields table 2.8, while for the case where a $40MHz$ bandwidth is used, we get table 2.9.

# jammers	1	2	3
v	64.26%	32.12%	0%

Table 2.8: Expected link throughput for 802.11n with $20MHz$ BW.

# jammers	1	2	3
v	5.06%	1%	0%

Table 2.9: Expected link throughput for 802.11n with $40MHz$ BW.

The results suggest that while immensely useful in terms of increasing the data rates under benign conditions, channel bonding can increase the vulnerability of a frequency hopping technique to jamming. More importantly, we observe that the limitations of frequency hopping as a jamming mitigation technique carry over to 802.11n networks.

2.8 Discussion

Our game theoretic framework can be applied with other variants of a jamming attack. As an example Xu *et al* [6] introduce the random and the reactive jamming model. With the former, the jammer transits between active and idle periods. Each of these periods follows a random distribution. A reactive jammer, senses the medium for ongoing communications, and whenever there is a legitimate packet on the air it jams the medium. The model presented in this chapter can be applied to account for these jamming strategies as well. In the following, we will present its application for the case of a random jammer.

Let us assume that the jammer picks its active periods T_a from a uniform distribution $\mathrm{U}[a, b]$ secs and its idle period T_i from the uniform distribution $\mathrm{U}[c, d]$ secs. Thus, the average active and idle times are $E[T_a] = \frac{a+b}{2}$ and $E[T_i] = \frac{c+d}{2}$ respectively. Consequently, the effectiveness of a random jammer is reduced by a factor α as compared to the case of the constant jammer, where:

$$\alpha = \frac{E[T_a]}{E[T_a] + E[T_i]} \tag{2.28}$$

Incorporating this factor, the corresponding payoff matrix for a single jamming device is now given by :

$$A_{i,j}^{1,a,rand} = \begin{cases} (1-\alpha) & \text{if } i = j, \\ 1 & \text{elseif } (i = 8 \text{ and } j = 9) \text{ or } (i = 9 \text{ and } j = 8), \\ (1 - 0.88 \cdot \alpha) & \text{elseif } |i - j| = 1, \\ 1 & \text{otherwise.} \end{cases}$$

Solving the game using the above payoff matrix will provide us with the solutions that correspond to the random jamming model.

2.9 Conclusions

In this chapter we seek to examine the effectiveness of FH as anti-jamming technique. We provide a game theoretic framework in order to capture the interactions between a link and a jammer employing FH. Our framework is measurement driven and accounts for two performance limiting factors; the number of available orthogonal channels as well as the adjacent orthogonal channel, jamming-interference. After formally presenting our framework, we show how we can apply it to 802.11 networks in order to quantify the efficacy of FH as jamming countermeasure. We conduct extensive experiments on our indoor wireless testbed in order to derive the payoff matrix of our

game. Our results indicate that frequency hopping is inadequate for protecting 802.11 networks from jamming with current spectrum allocations. We further validate our analytical results through experimentation with a prototype proactive FH scheme. We also show that with the same payoff matrix, if the number of orthogonal channels supported was much larger, frequency hopping would be very effective in coping with jamming. Finally, specific features of 802.11n, that is, channel bonding and carrier sensing, make it more susceptible to jamming attacks as compared to legacy systems, reducing further the efficacy of FH.

Chapter 3

Detecting Selfish Exploitation of Carrier Sensing in 802.11 Networks

Recently, tuning the clear channel assessment (CCA) threshold in conjunction with power control has been considered for improving the performance of Wireless LANs. However, CCA tuning can be exploited by selfish nodes (or alternatively, *active jammers*) in order to obtain an unfair share of the available bandwidth. In particular, by increasing the CCA threshold, a selfish client can manipulate the carrier sensing mechanism to ignore the presence of other transmissions on the medium; consequently, it increases the probability of accessing the medium and therefore obtains a higher, unfair share of the available bandwidth. In this chapter, we propose a novel approach to detect this misbehavior in WLANs. A key insight that leads to our approach is that a misbehaving node that has increased its CCA is unlikely to recognize low power receptions as legitimate packets; by intelligently sending low power probe messages, an AP can detect a misbehaving node with high probability. In a nutshell, our contributions are as follows: (a) We are the first to quantify the impact of selfish CCA tuning via extensive experimentation (b) We propose a novel lightweight scheme for detecting selfish nodes that inappropriately increase their CCA thresholds; we call our scheme CMD (for Carrier sensing Misbehavior Detection) (c) We perform extensive evaluations on an indoor 802.11 WLAN testbed to demonstrate that CMD detects misbehaving users with very high

accuracy (approximately 95 % of the time). Furthermore, it only incurs a false positive rate of less than 5 %.

3.1 Introduction

It is well known that the distributed coordination function (DCF) of the IEEE 802.11 MAC protocol provides long term fairness to the users that are in the proximity of one another and share the wireless medium [10]. Recently, there have been many approaches that advocate the joint tuning of the transmission power and the clear channel assessment (CCA) threshold to improve spatial reuse and thereby, the achievable capacity in a WLAN [44][45]. Tuning the CCA threshold opens the door for a new kind of selfish or malicious behavior. By increasing the CCA threshold, a "misbehaving" user[1] will cause the carrier sensing at the MAC layer to ignore the transmissions of other users with which, it shares the medium. As a consequence, (a) it may initiate transmissions when other transmissions are in progress thereby increasing collisions and, (b) it will not freeze its back-off counter while other nodes are transmitting packets; as a consequence it is able to access the medium much more frequently than other users and thus, enjoy a higher unfair share of the bandwidth. Given these adverse effects, it becomes critical that such misbehaving nodes are identified. *In this chapter, we propose a novel approach for detecting such nodes with high accuracy.*

There are two observations that drive our approach. **First,** a misbehaving node that increases its CCA threshold is likely to have a good "link" to the AP to begin with. If this is not the case i.e., the misbehaving node has a poor link to the AP, increasing the CCA can compromise the connectivity of the node; in other words, in lieu of gaining throughput, it will lose connectivity with the AP. **Second,** by increasing the CCA threshold towards gaining an unfair share of the throughput, the misbehaving node *implicitly* raises the bar with regards to the RSSI (Received Signal Strength Indicator) required for correct decoding. The receiver circuitry only tries to decode packets that are received with an RSSI that is higher than the CCA threshold. By increasing this threshold, packets

[1]We use the terms misbehaving, cheating, greedy and selfish interchangeably. We also use the terms user, node and client interchangeably.

should now be received with a higher RSSI value.

Based on the above observations, we design the Carrier sensing Misbehavior Detection (CMD) system. The key insight, evident from the above observations, is that a node that has increased its CCA threshold is unlikely to correctly recognize *low power transmissions* from the AP as legitimate packets. Thus, by sending low power probes, the AP can potentially detect such nodes with high accuracy. In order to reduce the overhead that will be incurred due to such probes, CMD first identifies the *set of possible* badly behaving nodes. This set consists of those nodes that are enjoying a significantly higher share of the throughput than their counterparts that are within the same cell. The probe messages are then only sent to the members of this set. Note here that, under saturated conditions where this problem is likely to be most critical, this set naturally excludes nodes that are at the periphery of the cell or nodes with poor links. Furthermore, as stated above, a node with a poor link is unlikely to be able to increase its throughput share via the malicious behavior considered.

In more detail our contributions in this chapter are as follows:

- We experimentally quantify the impact of selfish CCA tuning on the overall network performance. While previous studies have considered the benign use of CCA tuning to improve network performance, this is the first study that quantifies the extent to which, fairness suffers if this functionality were to be used inappropriately.

- We design and implement CMD for detecting such misbehaving nodes. CMD consists of two sub-components: (a) The Throughput Monitoring Module (TMM), which identifies a candidate set of possible misbehaving nodes and (b) The Low power Probing Module (LPM), which transmits the low power probes to effectively detect the *real* misbehaving nodes from among this candidate set. The implementation of CMD does not require any modifications to the IEEE 802.11 driver or firmware and can be implemented in the user space in its entirety.

- We analytically compute system parameters for CMD such that low *false positive* (wrongly classifying a well-behaved node) and *false negative* (not recognizing a misbehaving node)

probabilities are achieved. We validate our analytical results through measurements.

- We perform extensive experiments to evaluate CMD on an indoor WLAN testbed, with various configurations. Our experiments show that CMD detects misbehaving nodes with extremely high accuracy (95 %) with a very low false positive rate (< 5 %).

Our work in perspective: Selfish behaviors that target 802.11 functionalities have been considered and addressed previously. In particular, there have been many efforts that try to overcome behaviors where greedy nodes manipulate the back-off timers with 802.11 [46][47][48][49][50][51]. While a misbehaving node can enjoy lower back-off times by manipulating the CCA threshold (lesser chances of freezing the back-off counter), we wish to point out that the two attacks are not the same (as discussed later, the previously proposed strategies cannot deal with the considered attack). In particular, unlike the other attacks, tuning the CCA threshold is **protocol compliant**: the 802.11 standard [52] does not specify a value for the CCA threshold. In fact, different wireless network interface cards (NICs) have slightly different CCA thresholds. Although currently, tuning the CCA threshold is a functionality that these cards implement in the firmware, there are ongoing efforts towards enabling this functionality [53]. There have already been research efforts that advocate the tuning of this threshold for performance improvements [44][45]. In addition, GNU software defined radios [54][55] are expected to fully support 802.11 soon; such coexisting platforms that allow CCA tuning could be misused to pilfer a higher share of the throughput.

Finally, note that we only consider uplink traffic since one might expect that the APs, which are usually controlled by service providers, are unlikely or do not have the incentive to cheat; stated otherwise, it is unlikely that downlink traffic will be prone to such misbehaviors. The uplink traffic of a WLAN is not a negligible percentage of the total AP traffic anymore [56]. The increasing popularity of *p2p* applications (such as *bit-torrent*) result in a generation of a high proportion of uplink traffic in commercial hotspots.

The rest of the chapter is organized as follows. In Section 6.2 we discuss relevant CSMA/CA behavior in brief and related work. In Section 6.3 we describe our testbed at UC Riverside. Our

experiments to quantify the impact of the considered attack are presented in 3.4. In Section 3.5 we present the design and implementation of CMD; we analyze its performance in Section 5.3. In Section 5.5 we discuss the results of our evaluations of CMD. Miscellaneous issues are deliberated upon in Section 6.7. Our conclusions form Section 4.6.

3.2 Background and Related Work

In this section we provide a brief description of relevant CSMA/CA functions and describe related work.

Relevant 802.11 functions: 802.11's access policy is based on CSMA/CA. Each user needs to sense the medium idle for a specified time prior to transmitting data [18]. Whenever the perceived power on the medium is higher than the CCA threshold, a node must defer its transmission and enter the backoff state. Upon reaching this state, a node initiates a back-off counter with a random value. For each time slot that the medium is free, the counter is decremented; for each time slot that the energy on the medium is higher than the CCA threshold, the value of a counter is left unchanged (or *frozen*). When the counter value is decremented to zero, the node senses the medium again. If the power on the medium is lower than the CCA threshold (medium is idle) it transmits its packet; otherwise, it re-enters the backoff state; the expected counter value is now doubled.

When a misbehaving node increases its CCA threshold, it can result in the following effects:

- It can now ignore those signals that it *senses*, but are lower than this *new increased* threshold. Therefore, many of the signals on the medium have now no effect on transmission opportunities of the node.

- Other nodes that use the default CCA will sense the transmissions of the selfish node and will defer their own transmissions for longer periods.

- If a transmission of the misbehaving node is not successful, it will enter the backoff state. However, since its CCA threshold is increased, it is more likely that it will not have to *freeze*

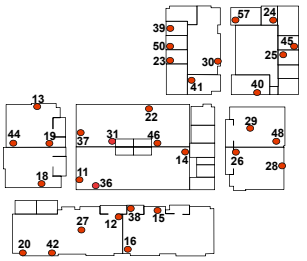


Figure 3.1: The UCR wireless testbed.

its backoff counter; this is a consequence of the node sensing the medium to be idle even if there are ongoing transmissions.

An interesting experimental study of how carrier sensing works in practice can be found in [57].

Related studies: While there have been prior efforts on overcoming attacks that manipulate 802.11 functionalities, the attack considered in this chapter has not received prior attention.

Attacks that violate the 802.11 back-off timers: Kyasanur and Vaidya [46] consider selfish behaviors where nodes deviate from the standard backoff mechanism of 802.11. They propose a mitigation scheme where the receiver explicitly assigns the backoff value to the sender. Konorski [51] proposes a misbehavior-resilient backoff mechanism. Gagalj *et al* [49] use game theory to develop a simple, localized and distributed protocol that guides multiple selfish nodes to a Pareto-optimal Nash equilibrium. Their work also considers attacks where nodes deviate from the backoff mechanism. Radosavac *et al* [47] present a framework based on Sequential Probability Ratio Test (SPRT) for detecting nodes that deviate from the backoff mechanism. Finally, Queseth [50] shows that it is hard to discourage selfishness by punishment if we cannot quickly detect these behaviors. All these studies however, are primarily related to the exploitation of the backoff mechanism, which is not the focus of our work. Note that in the considered setting, a node only increases its CCA threshold and does not violate the back-off policies; thus, these previously considered methods will not be effective.

Detecting other selfish behaviors: Raya *et al* [48] propose and implement DOMINO, a system for detecting various selfish behaviors in WLANs. DOMINO detects nodes that do not adhere to

the standard backoff mechanism, send out data without waiting for the standard DIFS period, use an oversized NAV to retain the medium for a longer time, or intentionally corrupt frames to get more medium access. In the attack considered, misbehaving nodes increase their CCA; none of the above behavioral trends are observed (as an example, the DIFS periods followed by the selfish nodes are legitimate). DOMINO cannot accurately detect an attack where nodes ”do not” freeze their back off counters due to ongoing transmissions. Consequently, DOMINO cannot detect possible CCA manipulations. Note that our approach can be complementary to DOMINO.

To the best of our knowledge we are the first to examine the selfish behaviors of clients in WLANs that try to increase their throughputs by exploiting the CCA threshold functionality.

3.3 Experimental Setup

In this section we provide a brief description of our testbed and the experimental methodology that is followed.

Testbed description: Our wireless testbed (Figure 3.1) is located on the 3rd floor of Engineering Building II at UC Riverside and consists of 32 Soekris net4826 nodes [33]; the nodes mount a Debian Linux distribution with kernel v2.6, over NFS. Each node is equipped with two miniPCI 802.11a/g WiFi cards, an *EMP-8602 6G* with Atheros chipset and an *Intel-2915*. We use the Mad-Wifi driver [34] for the *EMP-8602 6G* cards. We use a proprietary version of the *ipw2200* AP and client driver/firmware of the *Intel-2915* card. With this version we are able to tune the CCA threshold parameter.

Experimental methodology: For the purposes of our work we deploy the nodes of our testbed in a WLAN configuration (AP-client settings). The misbehaving clients exclusively use our Intel cards, since these cards allow us to tune CCA. The default value for the CCA threshold is -80dBm. All nodes use the maximum power (18dBm). Misbehaving nodes increase their CCA thresholds to the maximum value that guarantees association with the affiliated AP, while maintaining at least the throughput of the default settings (in isolation). We experiment with a large number of *config-*

urations; a configuration is a tuple: $< AP\ ID, Client\ List, Cheater >$. We provide more details on every experiment in the following sections.

3.4 The Problem

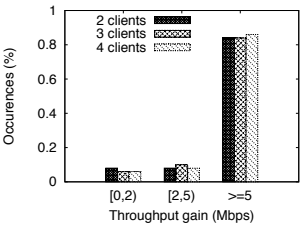


Figure 3.2: Increasing CCA can be an effective greedy strategy.

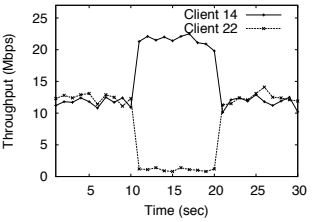


Figure 3.3: Time trace for the case of UDP traffic.

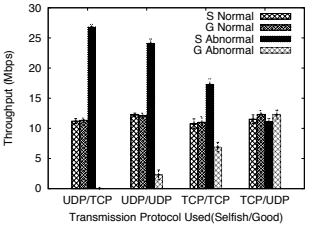


Figure 3.4: Impact of different transport layer protocols (S:selfish, G:good user).

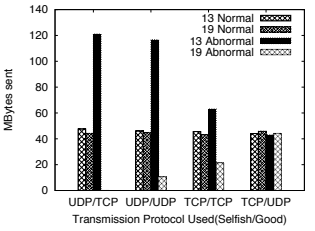


Figure 3.5: Bytes sent with different transport layer protocols (AP:44, S:13, G:19).

The 802.11 MAC protocol, as discussed earlier, provides long term *max-min* fairness to nodes that share a link. Under saturated conditions all the nodes that share a link, essentially access the medium with the same probability. By increasing the CCA threshold, a node can pilfer a higher share of the medium than it is entitled to, from the other users. To reiterate, transmissions that arrive at the receiver circuitry with an RSSI lower than the CCA threshold are ignored. By increasing the threshold, a node can ignore a significant fraction of the transmissions that occupy the medium. As described before, this not only causes increased collisions but also allows the misbehaving node to reduce the fraction of the time that it spends in the back-off state.

Our objective in this section is to demonstrate the effects of this greedy behavior via an extensive set of experiments on our testbed. We experiment with various configurations (with varying locations of the APs and clients) and measure the throughput gains of the selfish clients relative to their fair share of throughputs under normal operating conditions.

Experiments with saturated traffic: We depict our first results in Figure 3.2. The x-axis represents the throughput gains of the selfish clients and the y-axis represents the percentage of occurrences of this throughput gain (the gains are quantized into three levels); we vary the number of clients connected to the AP. We observe that in most cases (more than 85% of the 90 scenarios in total considered) the cheating user is able to gain significantly over the well-behaved clients affiliated with the same AP - at least 5Mbps gain from its fair share.

In some scenarios though (in fewer than 5% of the considered scenarios), the selfish client is unable to pilfer more than 2Mbps from the other clients. These cases arise when the selfish client is far from the AP (e.g., node 36 is the AP and node 22 is the selfish client) and as a result cannot increase its CCA to very high values; doing so would result in its disassociation from the AP. These studies suggest that a selfish node is likely to choose a location that is as close to the AP as possible[2].

In Figure 3.3 we present the temporal variations in throughput from a representative experiment. In particular, we use node 31 as an AP and nodes 22 and 14 as clients (Figure 3.1). We initiate fully saturated uplink traffic from both clients using *iperf* for 30 seconds. During the first ten seconds, both clients enjoy the same share of the throughput; this is a direct artifact of the fairness due to CSMA/CA. In the period between the 10^{th} and the 20^{th} seconds, node 14 misbehaves by increasing its CCA threshold from -80dBm to -50 dBm. We notice from Figure 3.3 that this results in a dramatic increase in the throughput of node 14. Meanwhile, node 22's throughput degrades significantly.

We observe that if the misbehavior is temporary, the effects are not long-lasting. As soon as the selfish user restores its default settings, the throughputs of the rest of the clients quickly return

[2]Note that well behaved users may also exhibit similar behaviors in order to improve the qualities of their links to the AP; such locations can result in higher RSSI values and thus, higher transmission rates can be sustained.

to the values under benign conditions. To understand this effect, recall that the selfish user follows the standard backoff mechanism with 802.11. After the settings are restored, within a short period of time, the greedy client enters the backoff state (senses energy on the medium). Then, the other users begin reducing their backoff counters; they gain access to the medium when their counter has reached the value of zero and at this point in time, fairness is restored[3].

Note also that even during the period where there is selfish behavior, the well-behaved nodes still obtain some throughput; this is directly attributed to the above reason i.e., packet losses can still occur for the misbehaving node and it can still enter the back-off phase.

Behavior with TCP traffic: In the previous experiments both clients 14 and 22 had fully saturated uplink traffic, i.e., they always had packets for transmission in their MAC layer queues. Next, we consider unsaturated traffic and in particular, TCP flows. The scenario again consists of well behaved clients and a greedy user that increases its CCA threshold.

The use of TCP results in two somewhat conflicting effects from the perspective of a well-behaved user. On the one hand, since the selfish user accesses the medium more often (as discussed above), the TCP packets experience longer delays and round trip times (RTT); thus, the TCP congestion window does not increase as rapidly as one might expect under normal operations and the overall throughput suffers. On the other hand, the selfish client itself might experience loss of packets and this causes its TCP connection to reduce its congestion window. In other words, packet losses have an impact on the data rate with TCP and thus, the selfish user will access the medium less often than it did in the UDP scenario.

In order to quantify the impact of CCA tuning on the behaviors of transport layer protocols, we conduct a large number of experiments. We use 90 different topologies using 15 different APs with 2 clients associated with them and consider all possible combinations of the two commonly used transport layer protocols, TCP and UDP. The misbehaving node employs its greedy strategy for the entire 30 second period (abnormal operation). The results are presented in terms of the average

[3]The speed with which this process occurs depends on the quality of the link between the AP and the misbehaving client. If this link is lossy, the misbehaving client is likely to experience a packet loss quickly and enter the back-off state.

throughputs of the well behaved and the selfish node in Figure 3.4; 95% confidence intervals are also shown. We also show the performance during normal operations where both clients are using the default settings. When the misbehaving client is sending UDP traffic its throughput gains are large. As one might expect, the impact is even higher when the well-behaved client is using TCP. The results show that significant gains are possible even if the link between the selfish user and the AP is lossy; this is because UDP does not reduce its sending rate upon experiencing packet losses. At the well behaved client, a lot of timeouts are triggered with TCP and the application throughput is extremely low (a few Kbps). When the misbehaving node uses TCP and the well behaved node uses UDP, the former is unable to achieve a significant gain in the throughput. This is a direct consequence of two factors (a) TCP regulates the sending rate thereby limiting the access opportunities for the selfish client and, (b) by increasing its CCA threshold, the selfish user can send more frequently, but when collisions are experienced its TCP source backs off whereas the UDP source at the well-behaved user does not reduce its rate. When both the well-behaved node and the selfish node use TCP, the latter benefits. Both TCP sources back-off when there are collisions; however, the selfish node is able to recover much faster since it is able to access the channel much more frequently. Figure 3.5 depicts the number of bytes sent per client for a representative configuration (AP-node 44, selfish client-node 13, well behaved-node 19).

To summarize, our experiments demonstrate that increasing the CCA threshold can lead to significant throughput benefits for the selfish client while hurting the other well-behaved clients, in a majority of the cases and with different transport layer protocols.

3.5 Detection system

In this section, we describe our scheme for detecting nodes that increase their CCA thresholds to gain an unfair throughput advantage in WLANs. We call our scheme CMD for Carrier sensing Misbehavior Detection system.

CMD is comprised of two sub-component modules: the first module, which we call TMM for

Throughput Monitoring Module, aims to identify the set of *potential* cheating clients; note here that this set consists of those clients that are suspected of cheating but may not necessarily be real misbehavers. The second module LPM (for Low power Probing Module) tries to identify the real misbehaving clients. The key insight that motivates the design of LPM is that nodes that have increased their CCA thresholds may not be able to correctly decode low power probes.

Client	14	22	37
Benign	9833	10521	10461
Cheating	320	521	21333

Table 3.1: TMM is effective with saturated traffic

Client	37	22	14
# Packets	1702	852	20322

Table 3.2: TMM can be mislead with unsaturated traffic

# Clients	3	5
Probing	26.1	21.8
No Probing	28.0	24.1

Table 3.3: Overhead with LPM

3.5.1 TMM: The Throughput Monitoring Module

As alluded to earlier, CMD sends probes in order to achieve its goal of detecting misbehaving users. Sending probes to all the clients associated with an AP can be prohibitive in terms of overhead. The goal of TMM is to identify the nodes that could be potentially cheating by increasing their CCA thresholds. Since the IEEE 802.11 is inherently fair, a node that gets a higher share of the available bandwidth could be a potential cheater. Note that it is not necessary that a node that gets a higher share of the bandwidth is essentially a cheater since different clients might have different traffic demands; the only conclusion that one can make is that such a possibility exists.

In order to identify the nodes that have a higher share of the medium, TMM monitors the volume of uplink traffic from each and every client. A node that is able to send a much larger volume of traffic is identified as a potential miscreant.

In order to demonstrate the effectiveness of this approach in terms of including misbehaving nodes in the set output by TMM, we perform the following experiment. We set up node 31 as an

AP and include 3 associated clients (nodes 14, 22 and 37); each client sends saturated traffic to the AP. We measure the number of packets transmitted from each client to the AP for a period of 10 seconds under two different scenarios: (a) when no client cheats and, (b) when client 37 cheats. The results are presented in Table 3.1.

These results suggest that monitoring the traffic can be effective in identifying misbehaving nodes. However, recall that in our experiments all clients have fully saturated uplink traffic. If the clients do not have saturated traffic they *may not* all have the same throughput under normal operations. In particular, if one of the clients produces a higher volume of uplink traffic, it will be mistakenly classified as a cheater if we were to just use TMM to identify the misbehaving nodes. To illustrate this we perform another experiment in which the same topology as in the previous case is used. The clients are now all benign. However, they have different application data rates: client 37 sends traffic at 2 Mbps, client 22 sends at 1 Mbps and client 14 at 24 Mbps.

Table 3.2 presents the results from this experiment. We observe that if TMM was used to classify nodes as cheaters, it would falsely conclude that client 14 is one. Thus, we need to further check if the nodes that are identified by TMM as potential cheaters are indeed cheaters or are legitimate recipients of higher throughputs; we do this using LPM (described later).

Implementation of TMM: We implement TMM in the user space. We develop a C application using `libpcap` [58]; the application is run at the AP and captures all the packets that arrive at its wireless interface. It internally maintains statistics in terms of how many packets are seen from each clients in a Z second time window (we will refer to Z as the *monitoring window size*). It then compares the number of packets from each client in order to identify the potential cheaters; if the number of packets that a client transmits, exceeds its fair share by X percent (we will refer to X as the *deviation value*), it is considered to be a possible cheater. We defer a discussion on how to choose the values of X and Z to Section 5.5.

When the potential cheaters have been identified, TMM calls LPM (described in the next section) to determine whether or not a "potential cheater" is *indeed* a "cheater". With this implementation we do not rely on an already available network monitoring system (for example, Ethereal

and tcpdump). Instead, it computes the statistics online. In Algorithm 1, we give the high-level pseudocode of TMM.

Data: IP addresses of the AP's clients

Result: A potential cheater

begin

1 Every Z seconds do:

2 **for** $i = 1$ *to* $num\_clients$ **do**

3 **if** $packets(i) > (1 + \frac{X}{100}) \cdot (\frac{total\_packets}{num\_clients})$ **then**

4 $Invoke\ LPM\ towards\ Client\ i$

end

end

end

Algorithm 1: Pseudocode for TMM

3.5.2 LPM: The Low Power Probing Module

The design of LPM is motivated by the observation that all the signals that arrive at the circuitry of a receiver with a received signal strength lower than the CCA threshold, are treated as noise; the receiver does not attempt to reconstruct packets from such signals [44]. Thus, a node that increases its CCA with the objective of increasing its throughput will not be able to correctly decode packets that are received with low powers. Thus, by having the AP probe the potential cheaters (determined by TMM) with low power packet transmissions, LPM achieves its goal of accurately identifying the real misbehaving clients.

A cheating node that increases its CCA towards obtaining a larger share of the available bandwidth, is likely to pick the *maximum* possible CCA without compromising on its connectivity with the AP[4]. The larger the CCA threshold, the higher are the number of possible ongoing transmissions that the carrier sensing logic ignores. If the CCA threshold is only increased slightly, the

[4]We assume this to be the selfish behavior for now; other possible variants are discussed in section 6.7.

selfish node will not be able to achieve significant performance gains. Note here that due to this very reason, it is unlikely that nodes that are either distant from the AP (or have poor quality links) will be able to effectively launch the attack under consideration; they will not be able to increase their CCA thresholds significantly without compromising their connectivity to the AP.

Design of LPM: The new CCA threshold (chosen by a selfish node) is based on the RSSI from the AP under default operating conditions. If the AP transmits with lower powers (as compared with default settings), the RSSI value at a receiver is reduced. Going further, if this transmission power is considerably lowered, packets may arrive at the misbehaving node's antenna with an RSSI that is smaller than its *increased* CCA threshold. This is the key idea that drives LPM. The AP, using a reduced transmission power, sends a probe packet to each client that has been flagged as a potential misbehaving client by TMM. If a client node has increased its CCA to the extent that it exceeds the RSSI of the received probe packet, the client node cannot respond to the AP. The latter waits for a preset period of time for the client's response; if no response is received, the AP flags the client as a misbehaving node. To reduce the possibility of false alarms, LPM challenges the potential cheaters (listed by TMM) with successive *ICMP_ECHO_REQUEST* packets (64 bytes), sent using a reduced transmission power. The client is expected to reply to each probing packet that is received from the AP. If more than $W\%$ of the reply packets are missing from a particular client, the AP declares the client as a misbehaving client. In Section 5.3 we discuss how we choose W and the probing power such that there is a good trade-off between the false positive rate and accurate detection with our system.

Data: Client i which has been flagged as a potential cheater by TMM

Result: Whether to declare it as a cheater

begin

1 $Ping(i, 10, Power_{probe})$

2 **if** *more than* $W\%$ *of reply packets are missing* **then**

3 $Declare\ Client\ i\ as\ a\ cheater$

end

end

Algorithm 2: Pseudocode for LPM

TMM reduces the probing overhead due to LPM. We point out that LPM increases the overhead by sending probe packets on the medium. If the AP were to probe all the clients, then the performance degradation could be significant, especially when the number of clients is large. Table 3.3 shows the degradation in the aggregate throughput of an AP when (i) all the clients had fully saturated uplink traffic and (ii) the AP was constantly probing the clients in a round robin fashion with 10 probe packets sent each client during a probe cycle.

We observe that if there are 3 clients associated with the AP the degradation is about 7 %; when there are 5 clients, the degradation is about 9.5%. As the number of clients increases, the degradation is higher; therefore, it is crucial to reduce the number of clients that LPM checks for real cheaters. Based on this, it is clear that TMM plays an important role in our system.

Note also that currently, we use the 64 byte *ICMP_ECHO_REQUEST* messages as probes; it is possible to reduce the overhead by creating special probe messages that are of smaller size. However, this will increase the complexity of the implementation (the current implementation is described below) and may require modifications to the 802.11 driver/firmware.

Implementation details of LPM: We have implemented LPM in the user space, on top of the wireless NIC's driver. It is run at the access point. Our implementation uses a shell script that invokes the *ping* application [59] to probe the clients. More specifically, the script consists of a loop which parses the list of clients that are flagged as potential cheaters by TMM. We set the

transmission power of the "ping" packets using the *iwconfig* command. Based on the results of the ping trials, LPM decides on whether a client is a cheater. This implementation is generic in that it can be run in conjunction with most commodity wireless NIC drivers.

For our Atheros cards, which use the MadWifi driver, we have also implemented our own probing utility using the Click Modular Router [60]. We use the `ICMPPingSource` and `ICMPPing Responder` elements to implement a *probe sender* and a *probe receiver*, respectively. The `SetTX Power` element enables us to set the transmission power for each $ICMP$ packet sent out by LPM. This element simply sets the Wifi TXPower Annotation flag on the packet to be sent, and we do not need to subsequently call *iwconfig* to set the power.

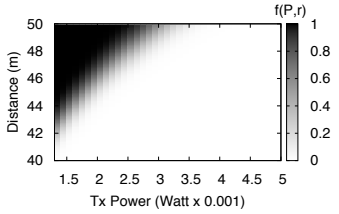


Figure 3.6: $f(P,r) = Pr\{signal(P,r) < CCA_{def}\}$.

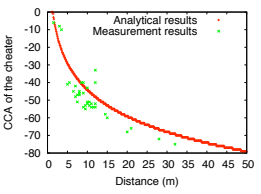


Figure 3.7: The theoretical and practical CCA_{cheat}.

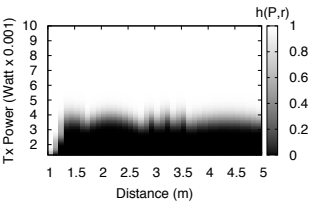


Figure 3.8: $h(P,r) = Pr\{signal(P,r) > CCA_{ch}\}$.

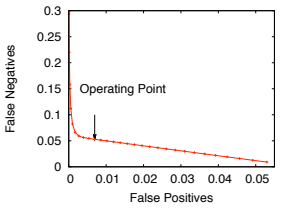


Figure 3.9: The analytical ROC curve of our system.

3.6 An Analytical Model to Derive System Parameters

The design of LPM is based on the observation that a cheating node with an increased CCA is unlikely to respond to probe packets sent by the AP with a low transmission power. There are two cases, however, where LPM may not lead to correct diagnosis: **(i)** Benign clients located at the border of the AP's coverage area may not be able to respond to low power probe packets sent

from the AP; these packets are likely to arrive at their circuitry with an RSSI lower than the default threshold CCA_{def}. This results in what we call *false positives*. (Note here that even though the links to such clients are likely to be poor, some of these clients may be getting a higher share of throughput in unsaturated traffic conditions). **(ii)** Misbehaving nodes could be so close to the AP that in spite of the AP using reduced transmission powers, probe packets can still reach their circuitry with an RSSI higher than their increased CCA value. In this case, the misbehaving node is not identified i.e., we have a *false negative*. In this section we analyze the performance of our system to determine various parametric inputs to CMD such that the false positive and false negative rates are kept low.

Propagation Model: In order to analytically determine the false positive and the false negative rates, we need to assume a propagation model. We calculate the received power P_r at distance r with transmission power P to be:

$$P_r = \frac{P}{r^\alpha} \cdot Y, \tag{3.1}$$

where α is the path loss exponent and Y is a random variable that is log-normally distributed. The random variable Y models the shadow fading effects and it has a mean value of one and a standard deviation equal to the shadow fading variation (obtained from measurements). The above model has been shown to be fairly accurate in indoor settings [61] [62].

False positives: We first compute the false-positive rate.

The probability $f(P, r)$ that a probe packet from the AP arrives at distance r with an RSSI below CCA_{def} is given by:

$$\begin{aligned} f(P,r) &= Pr\{\frac{P}{r^\alpha} \cdot Y < CCA_{def}\} = Pr\{Y < \frac{CCA_{def}}{P} \cdot r^\alpha\} \\ &= \frac{1}{2} + \frac{1}{2} \cdot erf(\frac{ln(\frac{CCA_{def} \cdot r^\alpha}{P}) - \mu}{\sigma \cdot \sqrt{2}}), \end{aligned} \tag{3.2}$$

where μ and σ are the parameters of the log-normal distribution (computed from the mean and the measured standard deviation). We plot this probability in Figure 3.6. In generating this probability,

the following values are used to derive the results: (i) CCA_{def}=-80dBm, (ii) the shadow fading variation is 5dBm (as measured from our testbed), and (iii) $\alpha = 5$, which is a typical value for the path loss exponent for an indoor environment [61] [63]. The figure shows that with extreme low power operations (1.5 mW), the probability of violating the default CCA threshold is extremely high (false positive); with moderately low powers (3 mW), this same probability is almost zero upto distances of 50 meters.

Equation (3.2) gives the probability that a packet arrives at the client's circuitry, after traveling distance r, with power less than CCA_{def}. Let us assume that LPM transmits 10 probe packets and expects n replies. Let $pr^{pos}(P, r, n)$ denote the probability that fewer than n probe packets[5] arrive at a distance r with an RSSI greater than CCA_{def}. This probability is given by:

$$pr^{pos}(P, r, n) = \sum_{k=1}^{n} (1 - f(P, r))^{k-1} \cdot f(P, r)^{10-k+1} \tag{3.3}$$

In order to calculate the false detection rate at distance r when the transmission power is P, we need a spatial distribution of nodes $s(r)$. As discussed in Section 3.4, nodes tend to stay close to the AP in reality. In order to get numerical results, a possible spatial distribution that can be used based on the previous observation is $s(r) = \frac{1}{ln(50) \cdot r}$, for $1 \leq r \leq 50\ m$ and zero otherwise[6] (the constant $\frac{1}{ln(50)}$ is chosen to assure that function s is a valid probability density function). With this spatial distribution model, the false positive rate $\pi^{pos}(P, r, n)$ at a distance r, when the transmission power is P is given by:

$$\pi^{pos}(P, r, n) = pr^{pos}(P, r, n) \cdot s(r) \cdot \Delta r|_{\Delta r \to 0} \tag{3.4}$$

We can then compute the overall false positive rate $\pi_p(P, n)$ when the AP is using transmission

[5]We assume that the channel is reciprocal and thus, if the probe message is correctly received, the corresponding $ICMP\_ECHO\_REPLY$ packet will be received with very high probability; this assumption ensures the tractability of our analysis.

[6]Nodes are expected to have a minimum distance -e.g. 1m - from the AP which in commercial hotspots are deployed mainly on ceilings. Note that our analysis can incorporate any other spatial distribution.

power P and when LPM expects n replies to its probes by integrating over the area of the cell:

$$\pi_p(P,n) = \int_0^\infty pr^{pos}(P,r,n) \cdot s(r)\, dr \tag{3.5}$$

False negatives: Similar steps as above are taken in order to compute the false negative rate. However, we first need to estimate the CCA threshold, that a cheating node at distance r is likely to use. The goal of the selfish client is to avoid as many transmissions as possible by increasing its CCA while maintaining its connectivity with the AP (note that this is when the AP is using the default power P_{def}, i.e., under default operations). The CCA chosen according to this strategy can be computed by solving the following optimization problem:

$$\text{maximize} \quad CCA_{cheat}(r) \tag{3.6}$$

$$\text{subject to} \quad Pr\{\frac{P_{def}}{r^\alpha} \cdot Y > CCA_{cheat}(r)\} = 1 \tag{3.7}$$

$$CCA_{cheat}(r) \in \{-80, -79, ..., 1, 0\} dBm \tag{3.8}$$

Solving the above optimization problem for various distances r, we get Figure 3.7. We present in the same figure the corresponding $CCA_{cheat}(r)$ (the CCA threshold tuned as per the same strategy), measured from our testbed; for a given location of the cheater, we increase the CCA threshold to the extent possible without compromising the connectivity with the AP. The results indicate that the analytical results match reasonably well with the measurement results; the coefficient of determination R^2 [64] is calculated to be equal to 0.71.

Having computed $CCA_{cheat}(r)$, we now proceed to calculate the false negative rate. We first calculate the probability $h(P,r)$ that a signal transmitted from the AP with power P arrives at

distance r with a RSSI greater than $CCA_{cheat}(r)$:

$$h(P,r) = Pr\{\frac{P}{r^{\alpha}} \cdot Y > CCA_{ch}(r)\} = Pr\{Y > \frac{CCA_{ch}(r)}{P} \cdot r^{\alpha}\}$$
$$= \frac{1}{2} - \frac{1}{2} \cdot erf(\frac{ln(\frac{CCA_{ch}(r) \cdot r^{\alpha}}{P}) - \mu}{\sigma \cdot \sqrt{2}}) \tag{3.9}$$

In Figure 3.8 we plot $h(P,r)$ for various AP transmission powers and distances from the AP (using the same parameters as previously) and $CCA_{cheat}(r)$ computed as the solution to the optimization problem defined in (3.6)-(3.8). We observe that if the cheater is extremely close to the AP ($\approx$ 1 m), there is no way of detecting it with low power probes. However, if the cheater is further than 1.5 meters, the use of a transmission power that is lower than say 3.5 mW can lead to an extremely high probability of detection, i.e., the probability that the signal is higher than the CCA set by the cheater is almost zero.

Given $h(P,r)$, we now calculate the probability $pr^{neg}(P,r,n)$ that no fewer than n packets arrive at distance r with an RSSI greater than $CCA_{cheat}(r)$:

$$pr^{neg}(P,r,n) = \sum_{k=n}^{10} h(P,r)^k \cdot (1 - h(P,r))^{10-k} \tag{3.10}$$

Using a spatial distribution of the nodes $s(r)$, we can calculate $\pi^{neg}(P,r,n)$, the false negative rate at distance r when the transmission power of the AP is P to be:

$$\pi^{neg}(P,r,n) = pr^{neg}(P,r,n) \cdot s(r) \cdot \Delta r|_{\Delta r \to 0} \tag{3.11}$$

Integrating over the whole area, we get the overall false negative rate $\pi_n(P,n)$ when the AP transmits with power P and LPM expects n responses to its probes:

$$\pi_n(P,n) = \int_0^{\infty} pr^{neg}(P,r,n) \cdot s(r)\, dr \tag{3.12}$$

Equations (3.5) and (3.12) provide the false positive and false negative rates of our system. These results also provide insights on the appropriate values for $Power_{probe}$ and n; these values should be chosen so as to satisfy a specific performance criterion. In short, we seek to minimize these probabilities; however, it is unlikely that they are both minimized together. Hence, we minimize the sum $\pi_p(P, n) + \pi_n(P, n)$. Solving this minimization problem yields $n = 9$ and $Power_{probe} = 3.3mW$. This means that in the LPM engine we need to set $W = 10\%$ (since 10 probes were set) and $Power_{probe} = 3.3mW$. In Figure (3.9) we present the ROC curve (Receiver Operating Characteristics) for the case $n = 9$ and we point out the operating point which corresponds to $Power_{probe} = 3.3mW$. Each point on this curve corresponds to a different $Power_{probe}$. Increasing $Power_{probe}$ increases false negatives; decreasing it will increase false positives. The operating point is the one that minimizes the aforementioned objective function. The corresponding false positive rate and false negative rates are: $\pi_p = 0.0053$ and $\pi_n = 0.054$. Note that with these settings, our detection system is able to achieve high detection accuracy.

3.7 Evaluation of CMD

In this section, we evaluate CMD.

Evaluation of the TMM module: First, we perform experiments to evaluate how TMM performs with various combinations of its input parameters; in particular, we consider the monitoring window size Z and the deviation $X\%$ from a client's fair share for it to be considered a potential cheater. Ideally, we want TMM to (i) flag all cheating nodes as potential cheaters *and* (ii) minimize the number of well-behaved nodes that are included in the set of potential cheaters. To evaluate the performance of TMM, we perform the following two sets of experiments.

(a) Monitoring legitimate traffic: In this set of experiments we monitor the traffic at the AP when no clients cheat and all clients have fully saturated uplink traffic. We vary both the monitoring window size Z and deviation $X\%$ from each client's fair share. The *false alert rate*, which represents the probability that a well-behaved client is flagged as a potential cheater, is depicted in

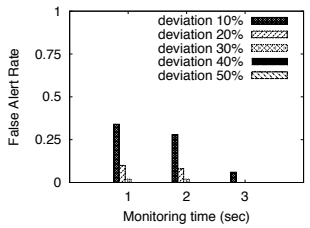


Figure 3.10: TMM false alert rate when there is no selfish user (2 clients).

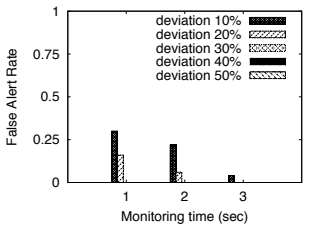


Figure 3.11: TMM false alert rate when there is no selfish user (3 clients).

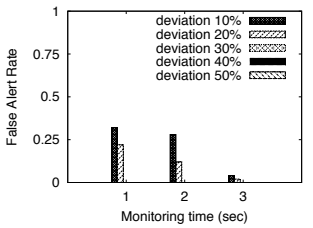


Figure 3.12: TMM false alert rate when there is no selfish user (4 clients).

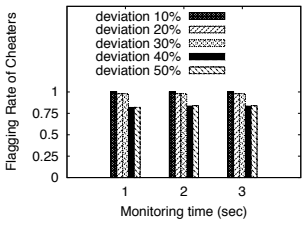


Figure 3.13: TMM flagging rate of cheaters when there is a greedy client (2 clients).

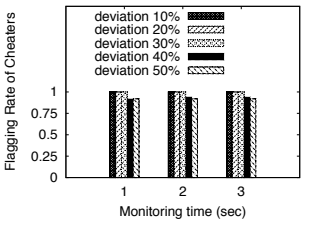


Figure 3.14: TMM flagging rate of cheaters when there is a greedy client (3 clients).

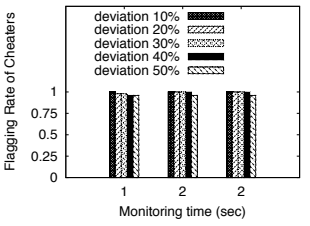


Figure 3.15: TMM flagging rate of cheaters when there is a greedy client (4 clients).

Figures 3.10-3.12; in these experiments, the numbers of clients associated with the AP are 2, 3 and 4, respectively.

From the results, we observe that when the deviation is chosen to be smaller than 20% the false alert rate can be very high, especially when the monitoring window size is small. For instance, when the deviation is set to 10% and the monitoring window size is 1 second, a well-behaved client is mistakenly flagged as a potential cheater with a likelihood of more than 30%. However, if we increase the deviation, the false alert rate decreases. When the deviation is set to 30% or higher, the false alert rate is very small. The results are somewhat expected since small deviations in the expected fair share of throughput are likely; furthermore, transients are possible if the monitoring window size is not sufficiently large. Reducing the false alert rate will reduce the overhead incurred due to probing with LPM.

(b) Monitoring the cheating nodes' traffic: In this set of experiments we monitor the traffic at the AP in the presence of cheating nodes. Again, all clients have fully saturated uplink traffic. In this case, we are interested in the false negative rates of TMM; in other words, we seek to measure the probability that TMM does not include a *real* cheater in its output set. Figures 3.13-3.15 depict

the probabilities that a cheating node is successfully identified as a potential cheater. From the results, we observe that when we use relatively small deviations (smaller than 30%) the TMM module almost always flags the cheating node as a potential cheater. If however it uses a deviation value that is higher than 30%, it misses the cheater in some cases.

The experimental results with both scenarios suggest that there is a tradeoff between the detection accuracy and the deviation value. Small deviation values help identify the cheating nodes but they may lead to high false alert rates under benign conditions; on the other hand, large deviation values help reduce the false alert rate but may miss some cheating nodes. In the current version of TMM, we set the monitoring window size to be 1 second and the deviation value to be 30%. Based on the experimental results, these values achieve a good balance between the false alert rate and the false negative rate of TMM.

Evaluation of the LPM module: LPM determines whether a potential cheater reported by TMM is indeed a cheater. We perform another set of experiments to quantify its detection accuracy. We experiment with a variety of configurations that take into account both saturated and unsaturated uplink traffic. In particular we experimented with 132 configuration tuples. We utilize `iperf` to generate uplink traffic. The cheating node always has saturated traffic (since as discussed earlier, a misbehaving client is expected to adopt a greedy strategy in exactly these scenarios) and misbehaves shortly after the initiation of the experiment (8-10 seconds approximately). Each experiment lasts for 1 minute. We vary the transmission power of the probe packets between 3, 4, and 5 dBm. Recall that our analysis in Section 5.3 suggests a probe power of $3.3mW$; this corresponds to approximately $5dBm$. We compute the false positive and false negative rates with the LPM module. Note that since LPM takes the output of TMM as its input, these rates are the false detection rates for the whole system, CMD (the output of LPM is the output of CMD). The results are presented in Table 3.4.

From Table 3.4, we note that LPM produces low false positive rates and low false negative rates in real experiments; even when the transmission power of the probe packets is varied, the maximum false positive and the maximum false negative rates are no higher than 4.5% and 6%, respectively.

$Power_{probe}$	False positive rate	False negative rate
5dBm	0.015	0.060
4dBm	0.015	0.030
3dBm	0.045	0.015

Table 3.4: Detection accuracy of LPM

We also observe the tradeoff between false positive rates and false negative rates as we reduce (or increase) the probing power; if we keep reducing the probing power, the false positive rate increases while the false negative rate decreases. From among the three probing powers we have used, the sum of false positive rate and the false negative rate is the smallest when $Power_{probe}$ is $4dBm$. This value is slightly lower than the one derived with the analysis in Section 5.3. The reason for this is that the assumed propagation model and its parameters (i.e., path loss exponent) or the spatial distribution of nodes $s(r)$ with the analysis, may not fit with the characteristics of our testbed with very high fidelity. Furthermore, in our analysis we focus on the performance of LPM, without considering the impact of TMM. It is hard, if not impossible, to model the interactions between the two modules accurately. This would require $s(r) \cdot \Delta r$ to include also the probability of TMM reporting a node, at distance r, as a potential cheater; this is difficult because it requires the knowledge of the traffic patterns of all clients (e.g., whether they send saturated traffic or not and their application data rates) at each location. *In spite of these limitations, note that the false positive rate and false negative rate analytically derived (i.e., π_p and π_n in Section 5.3) are very close to what is observed with experimental results on the real testbed.*

In our experiments LPM mistakenly declares a few well-behaved nodes as cheaters; this happens especially when some of the clients have unsaturated uplink traffic. As discussed in Section 3.4, clients far away from the AP cannot gain much by applying the considered selfish strategy because they cannot increase their CCA thresholds to a significant extent. In the presence of unsaturated traffic, some well-behaved clients that are far away from the AP are wrongly flagged as potential cheaters by TMM if their application data rates are higher than that of those that are closer to the AP. Consequently with LPM, the probe packets from the AP may reach these clients with a RSSI below $CCA_{def} = -80dBm$. Thus, these well-behaved clients are unable to recognize these packets and

send responses to the AP. However, our experiments demonstrate that such possibilities are rare given that the poor quality of the links to such clients limits the throughputs that they can achieve.

We observe that the false negative rate is about 6% when the transmission power of probe packets is $5dBm$. As we reduce this power, the false negative rate decreases significantly. For instance, when probe packets are transmitted at power $3dBm$, the false negative rate drops to about 1.5%. Interestingly, if we further reduce the transmission power of probe packets to $1dBm$, all cheating nodes are successfully reported as cheaters[7].

3.8 Discussion

Mitigating the effect of CCA exploitation: The goal of this chapter is to *detect* users that selfishly increase their CCA thresholds in order to get throughput gains. *Mitigating* the effects of such misbehaving nodes will be considered in the future; however, we deliberate on possible ways of overcoming the adverse effects of such cheaters. The simplest solution is to punish a cheating client by disassociating it completely from the AP. There are other mitigation approaches that are less harsh. As an example, the AP can choose to reduce its transmission power, which forces the cheating client to decrease its CCA threshold if it wants to communicate with the AP. Alternatively the AP may intentionally drive down the throughputs of such misbehaving clients. In particular, the AP could "not send" MAC layer ACKs to the cheating node for some of its frames. As a result, the cheating node has to back off with a larger contention window; this in turn, increases the opportunity of access to the other well-behaved nodes. Implementation of this approach is challenging because currently most commodity NICs implement MAC layer acknowledgements in the firmware.

Response to improved cheating strategies: In Section 5.3 we assume that a cheating node always chooses the maximum CCA threshold that guarantees its connectivity with the AP. This assumption is reasonable only if the cheating node is greedy to the maximum extent (the strategy enables the node to ignore as many transmissions as possible). If the misbehaving node knows that

[7]However we expect that such a low $Power_{probe}$ can lead to a high false positive rate.

CMD has been deployed, it might set a CCA threshold lower than that to evade detection. Note here that a less significant increase in CCA will have a lower impact on the network; thus, there is an inherent trade-off between the performance gain and the possibility of detection that the cheater has to consider. Note that it is still possible to detect misbehavior by further reducing the transmission power of the probe packets. This may lead to higher false positive rates. However, from Figure 3.9 we notice that even if we use the lowest transmission power considered, the false positive rate is still very low (relative to the specific spatial distribution).

3.9 Conclusions

In this chapter we identify a new, powerful selfish behavior in WLANs: a misbehaving client increases its CCA to improve its chances of accessing the medium. CCA tuning has been considered previously towards providing network wide performance enhancements; this is the first study that considers the misuse of this capability. With extensive experimentation on a real testbed, we show that such selfish behaviors can cause extremely unfair allocations of the wireless medium. We develop a detection scheme that we call CMD for Carrier sensing Misbehavior Detection. We mathematically analyze its detection accuracy. We also implement CMD on an indoor wireless testbed. Through experiments we demonstrate that CMD detects such selfish clients in WLANs with extremely high accuracy and with low false positive rates.

Chapter 4

FIJI: Fighting Implicit Jamming in 802.11 WLANs

The IEEE 802.11 protocol inherently provides the same long-term throughput to all the clients associated with a given access point (AP). In this chapter, we first identify a clever, low-power jamming attack that can take advantage of this behavioral trait: *the placement of a low-power jammer in a way that it affects a single legitimate client can cause starvation to all the other clients*. In other words, the *total throughput* provided by the corresponding AP is drastically degraded. To fight against this attack, we design FIJI, a cross-layer anti-jamming system that detects such intelligent jammers and mitigates their impact on network performance. FIJI looks for anomalies in the AP load distribution to efficiently perform jammer detection. It then makes decisions with regards to *optimally* shaping the traffic such that: (a) the clients that are not explicitly jammed are shielded from experiencing starvation and, (b) the jammed clients receive the maximum possible throughput under the given conditions. We implement FIJI in real hardware; we evaluate its efficacy through experiments on a large-scale indoor testbed, under different traffic scenarios, network densities and jammer locations. Our measurements suggest that FIJI detects such jammers in real-time and alleviates their impact by allocating the available bandwidth in a fair and efficient way.

4.1 Introduction

The proliferation of IEEE 802.11 WLANs makes them an attractive target for malicious attackers with jamming devices [7, **?**]. A jammer typically emits electromagnetic energy thereby causing: *(a)* prolonged packet collisions at collocated devices, and *(b)* packet transmission deferrals due to legitimate nodes detecting continuous medium activity. Hence, jamming attacks can lead to significant throughput degradation, especially when they intelligently exploit the properties of the MAC protocol in use.

In this chapter, we first identify a clever jamming attack where the jammer can not only hurt its intended victim, but cause starvation to other clients that are associated with the same AP as the victim. We call this attack the *Implicit-Jamming* attack. We design and implement FIJI, a cross-layer anti-jamming system to effectively detect such jammers and mitigate the impact of their attack.

The implicit-jamming attack: An inherent characteristic of the IEEE 802.11 MAC protocol is that under saturated traffic demands, an AP (access point) will provide the *same* long-term throughput to all of its affiliated clients [65]. If a client cannot receive high throughput from its AP for any reason (e.g. long-distance AP→client link or high levels of interference at the client side), the AP will spend a large amount of time serving this client at a low transmission bit-rate; this rate is determined by the rate adaptation algorithm in use. This will compel the AP to serve each of its other "healthier" clients (to which it can support higher transmission rates) for smaller periods. In other words, the AP does not distinguish between clients with low-SINR links and clients with high-SINR links; the long times taken to serve the former class of clients hurts the time available to serve the latter class of clients. This behavior is referred to as *the performance anomaly* of 802.11 [10] and is caused by the inherent design principles of the IEEE 802.11 MAC protocol (described in more detail in section 6.2).

The implicit jammer exploits this anomaly. To illustrate, consider the scenario depicted in Fig. 4.1. In this scenario: *(a)* all clients have high-SINR links with their AP in benign conditions, and

(b) a low power jammer is placed next to a particular client (say client C) such that it does not *directly* affect any other client of the AP. The jammer causes high levels of interference at client C and thus, most of the packets sent by the AP to C are not successfully received. This in turn causes the AP to reduce the transmission rate used to serve C (an inherent property of rate adaptation). As a result, the AP spends more time attempting to serve C, and this reduces the fraction of time that it provides to its other clients. Thus, the throughput of all the clients drops significantly due to the jamming of only client C. In other words, jamming a small subset of clients (even only a single client) implicitly affects all the clients that are affiliated with the same AP.

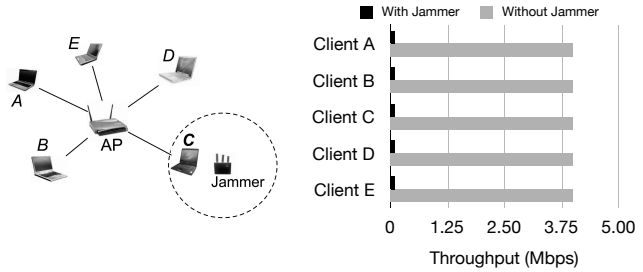


Figure 4.1: **Implicit Jamming:** The jammer takes advantage of the 802.11 performance anomaly. Using very low transmission power, it simply attacks client C. This is sufficient to tremendously degrade the throughput of all clients.

The impact of the implicit-jamming attack: In order to demonstrate the potential impact of this attack on the performance of the network, we conduct a set of preliminary experiments on our wireless testbed (described later in section 5.5). In particular, we construct the scenario in Fig. 4.1, where an AP maintains ongoing sessions with 5 clients and transmits saturated unicast traffic to all of these clients. We place a jammer 7 ft. away from one client (C). The jammer emits energy continuously at 0 dBm (1 mW), such that it causes interference to client C only. Fig. 4.1 depicts our throughput measurements, with and without the jammer. We observe that in the absence of jamming each client receives 4.1 Mbits/sec, on average. When the jammer is enabled, however, the long-term throughput of *all* clients drops to 90 Kbits/sec.

FIJI: An anti-jamming system to mitigate the implicit-jamming attack: In order to alleviate the effects of this intelligent attack, we design and implement FIJI, a distributed software system

that is executed locally at the APs. With FIJI, the AP is able to quickly detect an implicit jamming attack and identify the clients that are under the direct influence of the jammer(s). Furthermore, via a minimal set of online calibrating measurements that characterize the impact of the attack, the AP shapes the downlink traffic such that: *(a)* the jammed clients receive the maximum possible throughput given the circumstances, and *(b)* the rest of the clients are unaffected, i.e., shielded from the influence of the jammer(s). Some parts of FIJI are implemented on the *Click* software framework [60] and the rest are implemented on the driver/firmware of our wireless cards. Via extensive experiments, we observe that FIJI effectively mitigates the implicit-jamming attack on an 802.11a/g wireless testbed.

Our work in perspective: FIJI can be potentially applied in scenarios wherein jammers attack APs directly. However, in this work, we focus on addressing intelligent jammers that exploit the performance anomaly at the client side. Moreover, note that the impact of implicit jamming is exacerbated in downlink traffic scenarios; with uplink traffic, jammed clients will simply defer accessing the medium and will thereby allow the other clients to obtain higher levels of access.

The remainder of the chapter is structured as follows. In section 6.2, we provide a brief background on the performance anomaly in 802.11 as well as jamming attacks, and discuss related studies. In section 4.3, we describe the implicit jamming detection and mitigation with FIJI, our anti-jamming system. We describe the implementation of FIJI and evaluate its effectiveness in section 5.5. Section 5.6 provides the scope of our study. We conclude in section 4.6.

4.2 Background and Previous Work

In this section, we first describe the so-called performance anomaly with IEEE 802.11 and efforts related to addressing the anomaly. We then discuss jamming attacks in brief as well as prior work related to anti-jamming.

4.2.1 Performance Anomaly in 802.11 WLANs

Heusse et al. [10] were the first to observe that the long term throughput of all the clients associated with an AP in a WLAN is limited by the client with the poorest link. This effect eventually provides the same long-term throughput to all clients. Although [10] considers uplink traffic, this "anomaly" arises with downlink traffic as well [66, 45]. With either uplink or downlink saturated traffic, 802.11 provides equal medium access probability to all links. Let us consider the downlink scenario. An AP→client link with low SINR will coerce the rate adaptation mechanism at the AP to use a low transmission rate for this client. Thus, when attempting to serve this client, the AP will spend large amounts of time. Given that the AP will access the channel with equal probability for low-SINR clients and high-SINR clients (higher bit rate, shorter transmission durations), the latter will be served for smaller proportions of time.

Let us assume that AP α is sending saturated unicast traffic to each of its κ clients. The *theoretical* instantaneous transmission rate from AP α towards client c_i, where $i \in \{1, ..., \kappa\}$, is a step function of the SINR for this client [44]. In this work, we consider f_{c_i} to be the instantaneous *deliverable* rate towards client c_i, which in practice may not always be equal to the transmission rate (especially at high rates). Each client c_i of AP α will receive the ***same*** throughput T_i in the long term; this throughput is given by:

$$T_i = M_\alpha \cdot \frac{B}{\sum_{i=1}^{\kappa} \frac{B}{f_{c_i}}} = M_\alpha \cdot \frac{1}{\sum_{i=1}^{\kappa} \frac{1}{f_{c_i}}} \quad . \tag{4.1}$$

In the above equation, M_α is the fraction of the time that AP α is able to access the medium, given the contention with its co-channel neighbor devices. We assume that AP α transmits data packets of the same length B to all clients. From the above equation it is evident that if a client c_i receives low throughput, *all* clients will also receive equally low throughput under saturated conditions. Note that this phenomenon has been taken into account during the design of previous performance improvement algorithms for WLANs; examples can be found in [65], [66], [45], [44]. All these studies take the anomaly as a given and try to improve the network performance through other

intelligent strategies, such as AP load balancing and power control. In other words, such studies are inherently based on the fact that the 802.11 MAC protocol provides long-term fairness. Clearly, when this property of 802.11 is exploited by a malicious attacker, the performance of the schemes that are based on this property is also compromised. Hence, the existence of a mechanism that detects and mitigates such jammers becomes very vital.

Studies on mitigating the performance anomaly in 802.11: There have been numerous efforts on addressing the anomaly in 802.11. Most of them either require significant modifications on the 802.11 protocol functionality or they are very difficult to implement in practice.

Packet aggregation: Razafindralambo et al., [67] propose *PAS*, a technique that involves packet aggregation with dynamic time intervals. With PAS, nodes transmit consecutive packets back-to-back, separated by a SIFS period [52]. As a result, high-rate clients are able to transmit/receive many packets during an allocated time interval. However, packet aggregation requires modifications on the 802.11 protocol, in order to allow back-to-back data frame transmissions.

Contention window manipulation: Kim et al., [68] show that the anomaly can be addressed by tuning the 802.11 contention window size. They compute the minimum value of the window for the elimination of the anomaly. This technique, however, requires modification to the algorithm that selects the value of the contention window in 802.11. In contrast, our proposed scheme (described in the following section) does not require any changes to the 802.11 protocol semantics.

Data traffic manipulation: Bellavista *et al.*, in [69] propose *MUM*, an application-level middleware for facilitating multimedia streaming services. MUM tries to detect the anomaly by monitoring the RSSI of received packets and estimating the goodness of links. It employs the Linux `tc/iptables` to implement a hierarchical token buffer scheduler [70] that "differentiates" data transmissions towards low-rate nodes. The RSSI, however, cannot accurately capture the levels of contention and interference [71]. In addition, [69] uses a limited set of 4 static rate classes for traffic differentiation; *this setting is not adequate in jamming scenarios, as we show in section 5.5.* Along the same lines, Dunn *et al.*, [72] propose a heuristic for allocating a packet size to every client, which is proportional to the transmission rate. *We show in section 5.5 that the use of this heuristic during*

an implicit-jamming attack leads to some undesirable effects that in turn lead to poorer throughput than what is possible with FIJI. Similar approaches are followed in [73, 74] and [75]. Finally, Yang et al. [76] analytically model a WLAN with stations that support multiple transmission rates in order to demonstrate the performance anomaly. In contrast with these studies, our anti-jamming solution addresses the fact that the maximum transmission rate achieved by a single client can bound the total AP throughput. From the above discussion, as well as our measurements in section 5.5, it becomes evident that prior efforts on overcoming the performance anomaly problem in 802.11 cannot efficiently mitigate implicit jammers. We approach the 802.11 anomaly from the security point of view; in particular we examine a case where a malicious adversary can remotely exploit this feature as a vulnerability to cause complete starvation to the associated clients. *FIJI is effective against the implicit jamming attack, provides the best trade-offs between throughput and fairness and does not require any modifications on the 802.11 protocol.*

4.2.2 Jamming in Wireless Networks

Jammers are classified into two main categories based on their behaviors.

- ***Constant jammers:*** They emit electromagnetic energy all the time. This jamming technique is not usually adopted, since it depletes the battery of mobile jammers rather quickly. This category includes *deceptive* jammers [6], which transmit seemingly legitimate back-to-back data packets. With this, deceptive jammers can mislead other nodes and monitoring systems into believing that legitimate traffic is being sent over the medium.

- ***Intermittent jammers:*** They conserve battery life by emitting energy intermittently. As examples: (i) *Random* jammers alternate between random jamming and sleeping periods. (ii) *Reactive* jammers emit energy right after the detection of traffic on the medium, and remain inactive as long as the medium is idle. The implementation of reactive jammers is difficult; the detection and alleviation of such attacks is very challenging.

Previously proposed anti-jamming techniques: Prior work has focused on the impact of jamming on the performance of isolated wireless links. To the best of our knowledge, FIJI is the first system to examine the effects of implicit jamming on the *overall* performance of WLANs. Some previous studies employ frequency hopping techniques to avoid jammers [17, 1, 2]. We do not adopt such techniques in FIJI, since frequency hopping cannot overcome wide-band jammers [3], which are capable of jamming a plurality of the available bands simultaneously. Moreover, frequency hopping has limited effectiveness when multiple collocated jammers operate on different frequencies. FIJI, however, can be complementary to frequency hopping.

Gummadi *et al.* [17] show that even ultra-low power jammers can corrupt the reception of packets; towards coping with these jammers they propose a rapid frequency hopping strategy. Navda *et al.* [1] implement a proactive frequency hopping protocol with pseudo-random channel switching. They compute the optimal frequency hopping parameters, assuming that the jammer is aware of the frequency hopping procedure that is followed. Xu *et al.* [2] propose two anti jamming techniques: reactive channel surfing and spatial retreats. However, they do not consider 802.11 networks. In [6], efficient mechanisms for jammer detection at the PHY layer are developed. However, the authors do not propose any anti-jamming mechanisms. The work in [8] suggests that the proper adjustment of transmission power and error correction codes could alleviate jamming effects. However, it neither proposes an anti-jamming protocol nor performs evaluations of these strategies. Along the same lines, Lin and Noubir [77] present an analytical evaluation of the use of cryptographic interleavers with various coding schemes to improve the robustness of wireless LANs. In subsequent work, Noubir and Lin [78] investigate the power efficiency of a jammer. They show that in the absence of error-correction codes a jammer can conserve battery power by simply destroying only a portion of a legitimate packet. Finally, Noubir [79] proposes a combination of directional antennae and node-mobility in order to alleviate jammers.

None of these efforts consider the implicit jamming attack; FIJI is the first system to address this attack.

4.3 FIJI to Combat the Implicit Jamming Attack

In this section, we describe the design of our anti-jamming software system, FIJI. The goal of FIJI is twofold:

1. To detect the attack and restore the throughput on clients that are not explicitly jammed (we call these clients **"healthy"**).

2. To maintain connectivity and provide the highest possible throughput to clients that are explicitly jammed (we call these clients **"jammed"**).

FIJI involves the co-design of two individual modules, executed at the AP: a *detection* module and a *traffic shaping* module. We have implemented the two modules in the kernel space (we provide implementation details in section 5.5).

Attack model: In this work, we focus on low-power deceptive jammers. In particular, we assume that the jamming device has the following properties:

- It is placed next to legitimate clients. With this, the jammer is able to distort packets destined to the jammed client(s). In addition, the jammer is constantly transmitting packets back-to-back, thereby prohibiting the jammed clients from accessing the medium.

- It operates at very low power. As discussed earlier, the jammer simply needs to explicitly affect one of the clients of the AP. By transmitting at low power the jammer can conserve energy and make the detection of the attack a challenging task.

- It is able to operate on a wide band (covering all the available channels); this makes frequency hopping techniques inappropriate.

We describe the operation of the detection and the traffic shaping modules in what follows.

4.3.1 Detecting the implicit-jamming attack

The purpose of this module is to make the AP capable of detecting the jammed clients. *Previous jamming detection schemes assume that the jammed node is always the one that performs the detection. However with the implicit-jamming attack, the AP needs to detect the jammed client(s) in order to prevent the throughput starvation of the healthy clients.* As an example, in [6] the jammed node performs a consistency check between the instantaneous PDR (Packet Delivery Ratio), and the RSSI (Received Signal Strength Indicator) that it measures on its antenna. If the PDR is extremely low (i.e., almost zero), while the RSSI is much higher than the CCA threshold[1], the node is considered to be jammed. With the implicit jamming attack, however, the AP does not know the RSSI value that is observed by each of its clients. Thus, the approach in [6] does not allow the AP to detect the implicit jamming attack.

Measuring the transmission delay per client: FIJI relies on measuring the data unit transmission delay $d_{c_i} = B/f_{c_i}$ of every client c_i at the AP. More specifically, the denominator of Eq. (4.1) is the aggregate transmission delay D_α incurred by AP α in order to serve all of its associated clients once; it is the sum of the individual d_{c_i} values, $i \in \{1, ..., \kappa\}$, of the κ clients that are associated with AP α [65]. In other words, if we assume saturated downlink traffic, D_α corresponds to the average time that AP α needs in order to send one data unit to every client. The value of D_α is the same for all clients, and the transmission delay d_{c_i} of client c_i contributes to the value of D_α. Hence, a sudden, very large increment in D_α indicates that one or more of the d_{c_i} values has suddenly increased; *this would imply that one or more clients are under attack*. Towards calculating D_α, AP α needs to measure the d_{c_i} value for every client c_i (this includes possible retransmission delays and the rate-scaling overhead[2]). Measuring d_{c_i} will directly reveal the jammed clients: the value of $d_{c_i^J}$ for a jammed client c_i^J is likely to be much higher than the delays of the other clients. We adopt this detection strategy in FIJI.

[1]The CCA (Clear Channel Assessment) threshold specifies the RSSI value below which, receptions are ignored with regards to carrier sensing [44].

[2]The rate scaling overhead accounts for the higher delays incurred due to transient lower rates that the rate adaptation algorithm invokes.

4.3.2 Shaping the traffic at the AP to alleviate jammers

A trivial solution to the problem of mitigating the attack would be for the AP to simply stop serving the jammed clients. However, this would be unfair, since in many cases the jammed clients might still be able to receive data, albeit at lower rates. We opt to provide a **fair** bandwidth allocation solution; our twofold objective is to simultaneously achieve the following:

- **Objective 1:** For each of the healthy clients we seek to provide the same throughput that they would have enjoyed in the absence of the jammer, i.e., prior to the attack.
- **Objective 2:** A jammed client typically cannot receive much throughput as long as the jammer is active. Hence we want to provide to every jammed client the maximum possible throughput that it can receive, given that objective 1 is satisfied.

We refer to the state where these objectives are met as the ***optimal state***.

We propose a real-time, cross-layer software system to mitigate the effects of the implicit-jamming attack. The system is implemented partly in the Click module [60] and partly in the wireless driver/firmware. Click receives information from the MAC Layer with regards to the properties of the jammed clients. The AP→client traffic is then appropriately shaped and forwarded down to the MAC layer at the AP.

i) DPT: Controlling the data packet size: With this strategy, the AP fragments the packets destined to jammed clients; each such smaller fragment is now an independent packet. We call this approach DPT for *Data Packet Tuning*. With DPT, the rate at which these smaller packets are sent to the MAC layer is equal to the rate at which normal packets were forwarded to the MAC layer, prior to jamming. DPT is expected to have the following effects: *(a)* The transmission of small data packets is more robust to interference due to jamming; hence these small packets are more likely to be correctly deciphered by the jammed clients. *(b)* The rate at which the AP accesses the medium for the jammed clients remains unchanged; however, the channel occupancy time that is spent for

them is reduced, due to transmitting smaller packets to jammed clients. Hence, the AP will allocate a larger fraction of time for healthy clients.

Deriving the optimal data packet sizes: Our target is to determine the right packet size such that the optimal state is reached. The problem of achieving this state is formulated as follows.

Let us suppose that AP α has κ associated clients, and that n clients are being jammed, with $n \leq \kappa$. Our objective is to *minimize the aggregate transmission delay D_α^J of all the jammed clients c_i^J, $i \in \{1, .., n\}$ of AP α*. In other words, we seek to minimize

$$D_\alpha^J = \sum_{i=1}^{n} d_{c_i^J} = \sum_{i=1}^{n} \frac{J_i}{f_{c_i^J}} \ ,$$

where J_i is the data unit length for jammed client c_i^J, while $f_{c_i^J}$ is the deliverable rate at c_i^J.

Constraint: The $d_{c_i^J}$ value of each jammed client c_i^J must be at least equal (and as close as possible) to its data unit transmission delay d_{c_i} in benign conditions:

$$X1: \ \ d_{c_i}^J \geq d_{c_i} \Rightarrow \frac{J_i}{f_{c_i^J}} \geq \frac{B}{f_{c_i}}, \ \forall i \in n \ ,$$

where B is the default data unit length that the AP is using for all clients, and f_{c_i} is the deliverable rate to c_i^J in benign conditions. As explained earlier, the value of D_α is the same for all clients that are associated with AP α. If we sum constraint $X1$ over all jammed clients, the left hand side of the inequality is our objective function. With this we make sure that the healthy $\kappa - n$ clients will indeed experience an aggregate transmission delay very close to $D_\alpha = \sum_{i=1}^{\kappa}(B/f_{c_i})$; note that this is the aggregate transmission delay that was experienced by these clients prior to the jamming attack. Hence, by choosing the packet size J_i that results in a transmission delay that is as close to d_{c_i} as possible, we ensure that the throughput of the healthy clients remains unaffected (we elaborate on this later with an example).

Based on the above constraint, our optimization problem can be formulated as follows:

$$minimize: D_{\alpha}^{J} = \sum_{i=1}^{n} d_{c_i^J} = \sum_{i=1}^{n} \frac{J_i}{f_{c_i^J}} \tag{4.2}$$

$$subject\ to: \quad 1 \leq J_i \leq B,\ \forall i \in \{1, 2, ..., n\}, \tag{4.3}$$

$$and\ X1. \tag{4.4}$$

The solution to the above problem provides the values of J_i that minimize (4.2). Although the problem is an integer programming problem, it is easy to see that its special form ensures that it always has a solution, which can be found in polynomial time w.r.t. the number of variables.

How does DPT operate? Let us consider a case study with AP α, κ = 3, n = 1 and default packet size B. The transmission delays for the healthy clients c_1 and c_2 are d_1 and d_2, respectively; for the jammed client c_3, it is d_3. The long-term throughput of every client in benign conditions will be: $T_b = \frac{B}{d_1+d_2+d_3}$. If c_3 is now being jammed, its transmission delay will be $d_3^J > d_3$ and the new throughput will be: $T_J = \frac{B}{d_1+d_2+d_3^J}$. By applying DPT, the packet size towards c_3 will be J_3^{dpt} and its new transmission delay will be d_3^{dpt}. Since the rest of the clients are to maintain their old transmission delays (they are not explicitly jammed), the throughput with DPT will be: $T_{dpt} = \frac{B}{d_1+d_2+d_3^{dpt}}$. Our minimization problem ensures that $d_3^{dpt} \approx d_3$. Thus, for clients c_1 and c_2: $T_{dpt_1} = T_{dpt_2} \approx T_b$. In other words, DPT restores the throughput at the healthy clients.

Next, we show that the jammed client cannot receive a higher throughput if we further decrease the packet size[3] to a value $J_3^l < J_3^{dpt}$. With packet size J_3^{dpt} the throughput at c_3 will be: $T_{dpt_3} = \frac{J_3^{dpt}}{d_1+d_2+d_3^{dpt}}$. Let us assume that with packet size $J_3^l < J_3^{dpt}$ the transmission delay of c_3 is d_3^l. The throughput at c_3 will then be $T_{l_3} = \frac{J_3^l}{d_1+d_2+d_3^l}$. The required condition $T_{l_3} < T_{dpt_3}$ can be simplified as:

$$T_{l_3} < T_{dpt_3} \Leftrightarrow d_3^l > \frac{J_3^l}{J_3^{dpt}} \cdot (d_1 + d_2 + d_3^{dpt}) - d_1 - d_2.$$

[3]For larger packet sizes, objective 1 cannot be satisfied; hence we do not need to consider such a case.

Since the packet delivery rate f_{c_3} is the same, we have:

$$\frac{J_3^l}{J_3^{dpt}} = \frac{d_3^l}{d_3^{dpt}} \Leftrightarrow d_3^l = d_3^{dpt} \cdot \frac{J_3^l}{J_3^{dpt}}$$

$$\text{Thus: } \frac{J_3^l}{J_3^{dpt}} \cdot d_3^{dpt} > \frac{J_3^l}{J_3^{dpt}} \cdot (d_1 + d_2 + d_3^{dpt}) - d_1 - d_2 \Leftrightarrow$$

$$0 > (\frac{J_3^l}{J_3^{dpt}} - 1)(d_1 + d_2).$$

The last inequality is always true; hence, $T_{l_3} < T_{dpt_3}$.

Similar steps can be followed in order to show that DPT operates in the same manner in scenarios with multiple jammed clients. We adopt DPT in FIJI.

ii) DRT: An alternate approach. An alternative strategy would be to *explicitly tune the rate at which the packets are delivered* at the MAC layer (the packet size is now kept unchanged), destined to jammed clients. Fewer packets would arrive at the MAC layer for transmission towards the jammed clients, thereby allowing the AP to send traffic to healthy clients more frequently. Let us call this approach DRT for *Data Rate Tuning*. DRT operates as follows. Based on the measured d_{c_i} for each client c_i, the deliverable rate to every jammed client would be:

$$f_{c_i^J} = B/d_{c_i^J}. \tag{4.5}$$

DRT would bound the packet generation rate such that the data rate to the jammed client c_i^J is at most $f_{c_i^J}$. As a result, the rest of the (healthy) clients would share the remaining bandwidth. Thus, they would enjoy a share that is in fact higher than what they had prior to the attack. However, the packets destined to the jammed clients could be potentially lost due to channel or interference effects. Hence with DRT, the jammed clients will eventually receive *lower long-term throughput* than the specified (by DRT) rate of $f_{c_i^J}$. Clearly, while both DPT and DRT shape the traffic in order to overcome the implicit jamming effects, they essentially differ in the way they allocate the

bandwidth. With DPT the healthy clients receive the *same throughput as before the attack*, while the jammed clients achieve the *maximum possible* throughput under the circumstances. On the other hand, with DRT the healthy clients have a higher share of the bandwidth than in benign settings and receive *more throughput than before the attack*; the APs will spend more time serving the healthy clients, since most of the traffic is now destined to them. However, since the jammed clients do not reach their capacity, they are treated rather "unfairly". We evaluate this fairness versus throughput trade-off in section 4.4.3.

4.4 Implementation and Evaluation

In this section, we first describe our implementation of FIJI. Next we apply FIJI on a WLAN testbed and evaluate its efficacy in overcoming the implicit jamming attack.

4.4.1 The implementation of FIJI

FIJI is implemented entirely at the AP; no client software modifications are needed. In addition, FIJI does not require any special functionalities at the APs or at the clients; the only requirement is for the AP to be able to measure the d_{c_i} value for each affiliated client. Hence, FIJI can be applied on commercial APs through a driver/firmware update. In order to implement the two modules of FIJI we perform modifications on the driver and firmware of the AP, and we develop specific traffic shaping functionalities on the Click framework [60].

Implementing the implicit-jamming detection module: As explained in section 4.3.1, the AP needs to measure d_{c_i} for every client c_i. This will reveal, with high probability, the set of jammed clients. However, the value of d_{c_i} cannot be directly obtained from the driver of the wireless card; modifications in the firmware are required in order to compute this value. We use a prototype version of the *Intel ipw2200* AP driver/firmware; for every client we measure the time duration between the placement of the packet at the head of the MAC queue until an 802.11 ACK frame is received for this packet. The value is then passed up to the driver. The AP maintains a table in the

driver space with the d_{c_i} value for every client c_i. It also computes D_α^J (when jammers are active) and D_α (when jammers are inactive), by summing up the corresponding client delays. Temporary variations of the d_{c_i} values are handled by FIJI by using weighted moving average filtering; the previously maintained average is assigned a weight of 0.9 while the new sample has an associated weight of 0.1 (similar values are used in [65, 66]). Using these values, the AP constructs a table with the appropriate data packet sizes for the jammed clients. If the weighted $d_{c_i(new)}/d_{c_i(old)}$ value (for one or more clients) exceeds a pre-specified threshold δ, the AP computes the new packet sizes, updates the table and subsequently feeds it into the traffic shaping module, described below.

Implementation of the traffic shaping module: We implement the traffic shaper in Click. The module receives the table from the driver with suggested parameter settings for every client and shapes the traffic accordingly. We implement both DPT and DRT for comparison purposes. For DPT we have also developed an application-level script, which reads the table with the suggested packet sizes and inputs these values to the rude/crude measurement tool [80]. For DRT one may use two different Click elements, namely either the `BandwidthShaper(bandwidth)` or the `LinkUnqueue(latency, bandwidth)` element; we utilize the latter. Finally, we configure the AP to periodically flush the stored transmission delay values for every client and perform fresh delay measurements, using the default packet size. With this, we address scenarios of mobile jammers, which may move to the proximity of different clients, jammers with variable transmission power as well as jammers that stop operating.

4.4.2 Experimental set-up and methodology

Testbed description: Our testbed consists of 28 Soekris net4826 nodes [33], which mount a Debian Linux distribution with kernel v2.6 over NFS. The testbed is deployed in the 3rd floor of our campus building; the node layout is depicted in Fig. 4.2. Each node is equipped with an *Intel-2915* mini PCI WiFi card, connected to two 5-dBi gain external omnidirectional antennae. We use both the $main$ and aux antenna connectors of the card for diversity. As mentioned earlier, we use a proprietary version of the *ipw2200* AP driver/firmware of the *Intel-2915* card. With this version we are able to

(a) measure the D_α and D_α^J values at the AP, and (b) experiment with both 802.11a and 802.11g.

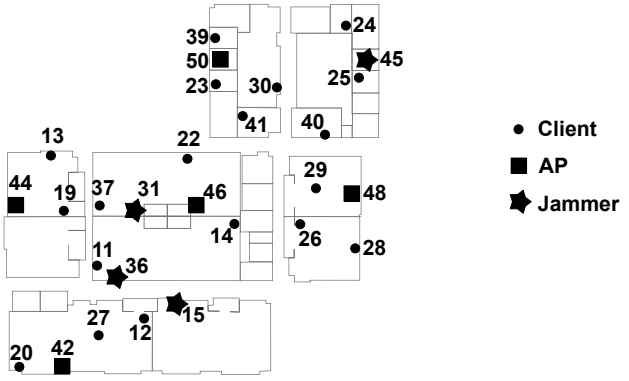


Figure 4.2: The deployment of our indoor 802.11a/g WLAN testbed in the 3rd floor of a campus building.

Constant jammer implementation: We have implemented our own deceptive jammer (instead of purchasing a commercial one [3]) since this gives us the freedom of tuning various jamming parameters. Our implementation of a constant jammer is based on a specific card configuration and a user space utility that sends broadcast packets as fast as possible. Our jammers are also equipped with the Intel-2915 cards; our ipw2200 prototype firmware for these cards allows the tuning of the CCA threshold parameter. By setting the CCA threshold to 0 dBm, we force the WiFi card to ignore all 802.11 signals during carrier sensing (packets arrive at the jammer's circuitry with powers much less than 0 dBm, even if the distances between the jammer and the legitimate transceivers are very small). The jammer transmits broadcast UDP traffic. This ensures that its packets are transmitted back-to-back and that the jammer does not wait for any ACK messages (the back-off functionality is disabled in 802.11 for broadcast traffic). We have developed an application-layer utility that employs *raw sockets*, allowing the construction of UDP packets and the forwarding of each packet directly down to the hardware.

Experimental methodology: For each experiment we first enable traffic from the AP to its clients and subsequently we activate the jammer(s). The duration of each experiment is 10 minutes; during each minute, the jammer is inactive for the first k sec, where $k \in [5, 20]$, and active for the other $60-k$ sec. We use a subset of 4 nodes as the jamming devices (nodes 15, 31, 36 and 45 in Fig.

4.2). We collect throughput and transmission delay (d_{c_i}) measurements once every 500 msec, for each client. We experiment with many different topological settings, with different numbers of APs and clients. By default all legitimate nodes set their transmission powers to the maximum value of 20 dBm and their CCA thresholds to -80 dBm. We examine both 802.11a and 802.11g links (unless otherwise stated, we observe the same behavior for 802.11a and 802.11g). The experiments are performed late at night in order to avoid interference from collocated WLANs, as well as not to cause interference to them. We use saturated UDP traffic with a default data packet size B = 1500 bytes. We also experiment with TCP traffic[4]. We use the *iperf* measurement tool to generate data traffic among legitimate nodes. We also use the *rude* tool to test DPT.

4.4.3 Does FIJI Deliver?

Next, we apply our anti-jamming framework on the testbed and evaluate its efficiency in alleviating the effects of implicit-jamming on the WLAN performance.

i) The efficacy of the detection module. We seek to observe two properties of this module:

1. *Efficiency of Detection*: How quickly can FIJI detect the presence of implicit jammers?
2. *Accuracy of Detection*: How accurately can FIJI determine if there is an ongoing jamming attack?

We conduct experiments with 5 APs and different numbers of clients with various link qualities. We configure the jammers to transmit at 0 dBm (1 mW) with CCA = 0 dBm, such that they affect one or more clients without affecting the APs.

a) On the speed of detection: Our measurements indicate that the transmission delay $d_{c_i^J}$ of a client increases sharply upon experiencing the implicit jamming attack. This increase is seen in less than 700 msecs; this time includes the transient periods before the weighted average $d_{c_i^J}$ converges

[4]The anomaly exists with TCP traffic as well [10]. Even though we do not present our TCP measurements, we observe that FIJI is similarly efficient with TCP traffic; we discuss this briefly in section 5.6.

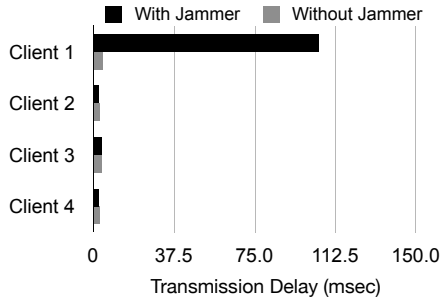


Figure 4.3: FIJI detects jammed clients by measuring their data unit transmission delays.

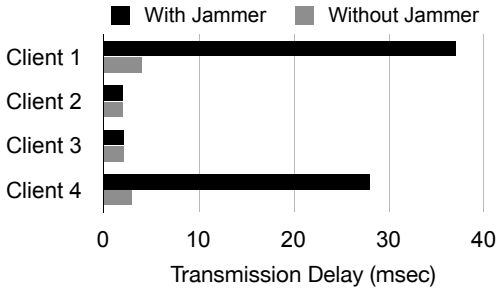


Figure 4.4: The jammer detection functionality of FIJI is accurate in most cases.

to a stable value. Fig. 4.3 depicts a delay snapshot with one AP and four clients with moderate-quality links. We observe that the $d_{c_1^J}$ value increases significantly (by 26 times in this experiment). Other experiments provided similar results. In summary, these results show that FIJI can quickly detect implicit jamming attacks.

b) On the accuracy of detection: We seek to evaluate FIJI in terms of its ability to detect an implicit jamming attack in the presence of interference. Note that the d_{c_i} value for a client c_i is affected by the levels of interference on the AP $\rightarrow c_i$ link. The higher the level of interference, the higher the d_{c_i} value. In order to evaluate this ability of FIJI, we perform experiments with multiple overlapping cells (each with its own AP), so that some clients suffer interference from one or more APs; in this setting, we activate our low-power jammers.

Detecting jamming on good quality links: We first consider links that have a high SINR. Fig. 4.4 depicts sample experimental results. In the snapshot of Fig. 4.4, a jammer is placed such that it affects 2 out of the 4 clients of an AP. We observe that FIJI is able to perform a successful detection. In general, our empirical observations suggest that when threshold $\delta \geq 9$, FIJI can effectively detect the attack (Fig. 4.4). In the experiment described above, the value of δ was 9.

FIJI and poor quality links: With poor quality links (SINR is low), FIJI cannot easily decide if a client is under attack or not. This effect is captured in Fig. 4.5, where the jammer affects a very poor link. In particular, the link 46→25 is considered with the node 45 acting as a jammer (Fig.

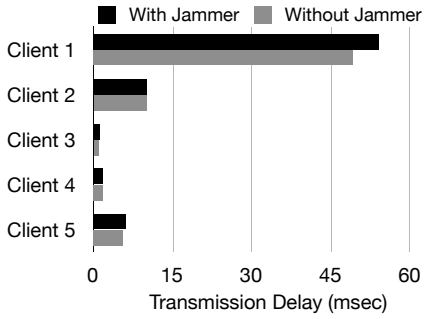


Figure 4.5: The jammer detection with FIJI is less accurate in scenarios with very poor links.

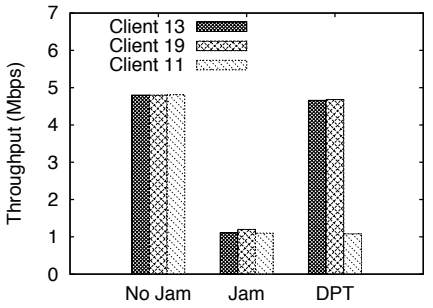


Figure 4.6: DPT restores the performance of healthy clients to that in benign settings.

4.2). The link achieves 190 Kbits/sec in the absence of jamming and 164 Kbits/sec under jamming. Since the jammer does not significantly increase the delay experienced on such poor links, FIJI cannot decipher whether the increased $d_{node-25}$ value is due to jamming or legitimate interference. However, in such conditions, the overall change in the network performance due to the jammer is unlikely to be significant; the presence of the poor link already hurts the network performance. Furthermore note that a jammer is unlikely to attack such poor quality links if it aims to harm the network to the extent possible.

In some extreme cases, a poor quality link (exposed perhaps to other interfering APs that are hidden from its own AP) might cause a client to experience large delays. In such scenarios with healthy but poor-quality links, FIJI may incorrectly classify such links as being *jammed*. Classifying such cases as attacks, though, is perhaps appealing in terms of improving performance for the rest of the network.

FIJI and high power jammers: An implicit-jamming attacker is likely to place its jammer(s) very close to one or more clients so as to:

- degrade the client's observed SINR value to the extent possible, and
- use a very low transmission power, in order to conserve energy and avoid detection.

As our experiments indicate, under these conditions, FIJI can identify the jammed clients in real

time since all measured $d_{c_i^J}$ values are usually extremely high for those clients. In contrast, a jammer could use high transmission power (although this could increase the chance of its detection and result in high energy consumption). Such a high power jammer is likely to affect multiple clients and even the AP itself, directly. The delays of all these clients may go up and in this case, given its design principles, FIJI may not be able to detect the jammer. However, there are other jammer detection techniques that can be used in conjunction with FIJI to detect such jammers [6].

ii) The traffic shaping module in action. Next we evaluate the efficacy of DPT and compare it against DRT.

DPT is the most fair solution: In a nutshell we observe that as long as the jammer is successfully detected, DPT restores the throughput at the healthy clients. A sample case is depicted in Fig. 4.6. Here, AP 44 transmits unicast traffic to clients 11, 13 and 19; node 36 is jamming client 11. In the absence of jamming each client receives 4.8 Mbits/sec on average. When the jammer is active, without enabling DPT, all clients receive 1.1 Mbits/sec on average. The solution to the problem formulated in (4.2) suggests that J_{11} should be set to 345 bytes. When DPT is enabled and this packet size is chosen for the jammed client, we observe that the throughput of the healthy clients 13 and 19 is restored to 4.66 Mbits/sec, while the jammed client 11 achieves about 1.1 Mbits/sec. Note that the healthy clients do not achieve their jamming-free throughput of 4.8 Mbits/sec. This is because in our solution the equality in the constraint $X1$ is achieved for a non-integral value of J_{11}; we round the value of J_{11} up to the nearest integer. With this, the transmission delay for the jammed client is a bit higher as compared to the delay under benign conditions and this slightly degrades the throughput at the healthy clients.

In order to validate that DPT provides the most fair bandwidth allocation, we experiment with many different J_{11} values. Fig. 4.7 depicts the results that correspond to the settings with two J_{11} values: 166 and 700 bytes. We observe that:

- With packet sizes smaller than J_{11}^{dpt} (case with 166 bytes), the jammed client does not reach its capacity (receives 360 Kbits/sec) and the AP spends more time serving the healthy clients

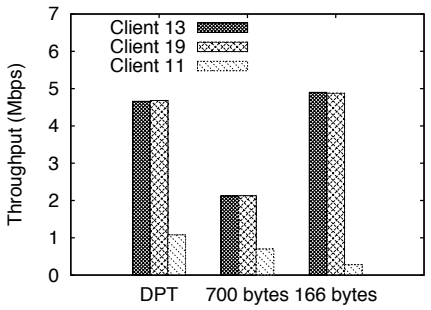


Figure 4.7: DPT always manages to provide a fair allocation of throughput among clients .

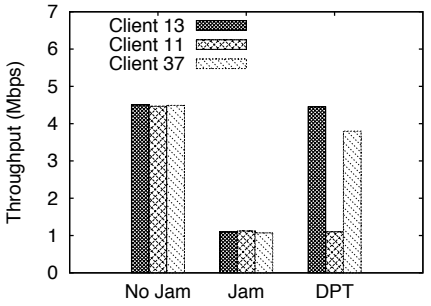


Figure 4.8: DPT can easily handle scenarios with multiple clients that are simultaneously jammed.

(as discussed in section 4.3): each healthy client now receives 5.1 Mbits/sec. Note that the value J_{11} = 166 bytes is computed using the approach proposed in [72] for the considered scenario and *it clearly does not provide the desirable fairness in terms of throughput*.

- When the packet size is higher than J_{11}^{dpt} (case with 700 bytes), the throughput at the jammed client is lower than 1.1 Mbits/sec; the healthy clients also underperform. This is again conformant with our analytical assessments in section 4.3 with regards to the maximum achievable throughput.

Multiple jammed clients: We have so far considered scenarios wherein a single client was jammed. Next, we examine scenarios with multiple jammed clients per AP. Our experiments reveal that DPT is also able to effectively mitigate the implicit jamming attack in such scenarios. Fig. 4.8 presents a sample case with AP 46 and clients 11, 37 and 14; the jammer-node 36 explicitly affects both clients 11 and 37. Under benign conditions all clients receive approximately 4.5 Mbits/sec on average. As soon as the jammer is activated, without enabling DPT, all clients receive about 1.1 Mbits/sec. DPT sets the value of J_{11} to be 367 bytes and J_{37} to be 1266 bytes. With this, *DPT is able to restore the throughput at the healthy clients*.

DPT vs. DRT: Using the same methodology, we examine the effectiveness of the DRT solution. Our measurements demonstrate that DRT provides much higher throughput to healthy clients. On

the other hand, DRT results in an additional unfair degradation at the jammed client. Fig. 4.9 represents the behaviors in an example scenario, with the same topological configuration as before (AP 44, clients 11, 13 and 19, jammer 36); the figure depicts the throughput prior to the attack (benign settings), with the jammer without DRT, and after the application of DRT. We observe that DRT overcomes the implicit impacts of the attack. Upon enabling DRT, clients 13 and 19 are no longer affected by the jammer and they receive 5.12 Mbits/sec each. Although DRT sets the maximum allowable data rate towards client 11 to be 1.1 Mbits/sec, the observed throughput at this client is significantly lower i.e., 680 Kbits/sec on average. This behavior of DRT conforms with our discussion in section 4.3.2; we observe similar trends in all our measurements with one or more jammed clients. *To summarize, with DRT the healthy clients receive more throughput than before the attack; however the jammed clients are penalized further.*

The choice between DPT and DRT depends on the performance objectives; one has to decide between fairness (with DPT) and bandwidth utilization (with DRT). DPT is fair: the healthy clients receive the same throughput as before the attack, while the jammed clients achieve the maximum possible throughput under the circumstances. On the other hand, DRT increases the throughput at the healthy clients and potentially, the total network throughput. However, the jammed clients cannot receive the maximum throughput that they can achieve in the presence of the jammer.

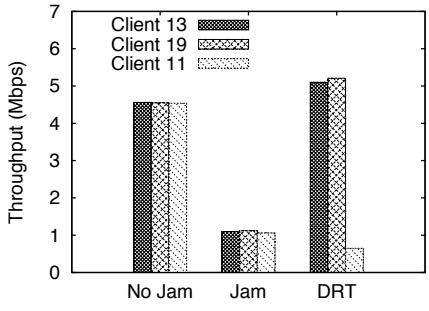


Figure 4.9: With DRT healthy clients receive more throughput than before the attack.

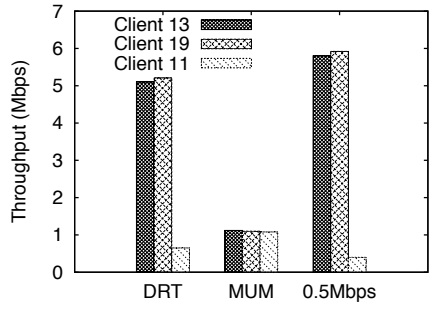


Figure 4.10: DRT satisfies our objectives better than other data rate allocation approaches.

Note that DRT also relies on the online measurement and use of d_{c_i}. With this, DRT seeks to

eliminate the effects of implicit jamming at healthy clients, while at the same time not degrade the throughput at jammed clients. Fig. 4.10 depicts a case with 802.11a where DRT sets the data rate at 1.1 Mbits/sec, while MUM [69] (recall our discussion in section 6.2) sets 6 Mbits/sec. We observe that by using data rates higher than the one chosen by DRT, the healthy clients are still affected by the attack, since in this case the downlink traffic for the jammed client is still saturated. Moreover, if we use lower data rates than the one chosen by DRT, the healthy clients get more service time, however the jammed clients receive much lower throughput than with DRT.

4.5 The Scope of Our Study

FIJI and previous studies on traffic shaping: Our work is the first to analytically derive the *optimal* settings for traffic shaping at the AP to mitigate the implicit-jamming attack. Traffic shapers have also been previously proposed in [69, 73, 74, 72]. Clearly, FIJI could also be considered as another traffic shaper, simply to overcome the performance degradation due to the 802.11 anomaly. Unlike FIJI however, previous traffic shaping schemes cannot overcome the effects of an implicit-jamming attack, as explained in sections 6.2 and 5.5. Other schemes that provide fair access to the WLAN resources [**?**, 65] would also be inadequate in combating an implicit-jamming attack since they are not designed for this purpose.

FIJI versus power control: Power control has been suggested as a means of mitigating legitimate interference [45, 81]. Typically with power control, nodes tune their transmission power and CCA settings in order to reduce the amount of interference from/to their neighbors. However, if the jammer is very close to one or more clients, its signal cannot be ignored through CCA adaptation. If a client increases its CCA threshold to a high level (to ignore the jammer's signal), the connectivity to the AP will be lost.

Addressing random and reactive jammers: FIJI can mitigate the interference due to any type of jammer, even random or reactive jammers. With prolonged random jamming and sleeping periods (order of seconds), FIJI can perform a rapid detection and then customize the data packet size,

as per the observed data unit transmission delay $d_{c_i^J}$. If the sleep and active periods of the random jammer are of the order of milliseconds, FIJI can monitor the *average* $d_{c_i^J}$ value instead. FIJI is expected to alleviate reactive jammers, too, since it only needs to monitor the impact of reactive jamming by measuring $d_{c_i^J}$. We have not experimented with reactive jammers, since implementing such a jammer is a very difficult task.

FIJI against other attacks: The two modules of FIJI can arguably be effective against any attempt to exploit the 802.11 performance anomaly in order to degrade the client throughput. As examples, a *compromised* device x could *deliberately* decide to *(a)* associate to a very distant AP α, or *(b)* accept traffic at a very low reception rate only (e.g. by discarding a large volume of correctly received packets). In both cases, x would receive a few Kbits/sec. Note here that, legitimate, non-compromised devices would follow such an approach only if they cannot associate with a better APs. However, given that (a) dense deployments of WLANs make the presence of an AP with a good quality link likely [45], and (b) distant poor quality APs are likely to be beyond the administrative domain of the client (the client will not be able to associate with such APs), the possibility of this is small in practice. FIJI can arguably be effective against such attacks. In particular, FIJI considers such clients to be jammed clients and ensures that the other clients remain unaffected.

FIJI and TCP: FIJI is implemented above the 802.11 MAC and below the transport layer at the AP. We have done measurements with TCP, which have demonstrated that: *(a)* Without FIJI, the performance anomaly also exists with downlink TCP traffic. The TCP packets that are destined to the jammed clients require a significant amount of time for successful delivery. As a consequence, the healthy clients are affected; they do not achieve the same throughput as before the attack. *(b)* Our experiments also demonstrated that the application of FIJI in TCP traffic scenarios is beneficial. By reducing the rate at which packets are delivered to the MAC for the jammed clients, DPT shapes the TCP traffic in a way that the healthy clients are unfettered. Note that the packet fragmentation with FIJI is executed after any TCP layer fragmentation; hence, FIJI does not intervene with TCP operations.

4.6 Conclusion

In this chapter we identify a low-power jamming attack that we call the *implicit jamming attack*. With this attack, a jammer exploits a performance trait of the IEEE 802.11 MAC protocol to cause starvation to not only an explicitly jammed client, but all the clients associated with the same AP as that client. Since the 802.11 MAC provides long term fairness (under saturation conditions) to the associated clients in terms of equal throughput, the attacker can nullify the AP throughput by affecting only one or at most a few clients.

We design, implement and evaluate FIJI, a cross layer software system for mitigating the implicit-jamming attack. FIJI is comprised of two modules, for detecting such an attack and shaping the traffic appropriately in order to alleviate the jamming effects. We evaluate FIJI on an 802.11a/g testbed, and under many different jamming scenarios. We show that FIJI can quickly detect the attack and effectively restore the throughput at the implicitly affected clients. FIJI also ensures that the jammed clients get as much throughput as they can under the circumstances.

Chapter 5

Lightweight Jammer Localization in Wireless Networks: System Design and Implementation

Jamming attacks have become prevalent during the last few years, due to the shared nature and the open access to the wireless medium. Finding the location of a jamming device is of great importance for restoring normal network operations. After detecting the malicious node we want to find its position, in order for further security actions to be taken. Our goal in this chapter is the design and implementation of a simple, lightweight and generic localization algorithm. Our scheme is based on the principles of the gradient descent minimization algorithm. The key observation is that the Packet Delivery Ratio (PDR) has lower values as we move closer to the jammer. Hence, the use of a gradient-based scheme, operating on the discrete plane of the network topology, can help locate the jamming device. The contributions of our work are the following: **(a)** We demonstrate, through analysis and experimentation, the way that the jamming effects propagate through the network in terms of the observed PDR. **(b)** We design a distributed, lightweight jammer localization system which does not require any modifications to the driver/firmware of commercial NICs. **(c)** We implement and evaluate our localization system on our 802.11 indoor testbed. An attractive and

important feature of our system is that it does not rely on special hardware..

5.1 Introduction

The widespread proliferation of 802.11 wireless networks makes them an attractive target for saboteurs with jamming devices [7, 3]. Numerous jamming attacks have been reported in the recent past [14, 15, 16, 4]. A jamming device continuously emits electromagnetic energy on the medium. The effect of this behavior on a CSMA/CA network is twofold: **(a)** at the transmitter side it renders the medium busy resulting in large back-off times and, **(b)** at the receiver side it dramatically decreases the SNR resulting in a large number of packet collisions. Note that jamming effects may also occur due to accidental activation of devices that do not serve a malicious cause, such as microwave ovens, cordless phones [82], etc. Following the detection of the presence of an attacker [23], an algorithm is needed for localizing the jammer, so that further countermeasures can be taken by the network (such as deactivating the jamming device, as well as isolating the attacker, capturing, punishing or even destroying it).

In this work, we design and implement a simple, low-overhead algorithm for jammer localization. The main attribute of our algorithm that makes it attractive to use and straightforward to implement, is that it relies on packet delivery ratio (PDR), a metric that is readily available at each node and is an indication of transmission corruption. Our technique exploits an intrinsic characteristic of the wireless medium: since the power of the jamming signal degrades with distance, farther transmitters do not sense strong jamming signals. In addition, the SNR requirement at such tranceivers is often satisfied. This cannot be concealed by the attacker. The transmitter is thereby able to send more packets, while the receiver can decode more of those, resulting in an increased PDR as we move away from the jammer.

Taking this property into account we design a decentralized localization algorithm based on the gradient descent minimization method. Our algorithm progresses in a distributed manner towards the proximity of the attacker by successive forwarding of PDR measurements to neighbors. In that

sense, it is reminiscent of the iterative gradient descent algorithm for identifying the minimum of a real-valued function f. This algorithm moves from one point $\boldsymbol{a}$ of the function's domain S to another $\boldsymbol{b} \in S$. The point $\boldsymbol{b}$ is towards the opposite direction of the gradient of f at $\boldsymbol{a}$; this is the direction in which f exhibits the largest decrease with regards to its value at point $\boldsymbol{a}$. If the algorithm cannot proceed further, (at least) a local optimum is declared. Note that in our case, the domain set is the discrete locations of the nodes. Hence, our scheme can be viewed as a discretized version of a gradient descent algorithm.

Our main contributions in this chapter can be summarized as follows:

- **Analytical and experimental assessments for the spatial characteristics of jamming effects in a network:** As previously mentioned, the jammer may affect both the transmitter and receiver operations; this has an impact on the PDR. We provide an analytical expression for quantifying the change in PDR in the different parts of the network (relative to the jammer's location). We experimentally validate the analytically computed expression on our testbed. We show that tranceivers further from the jammer exhibit lower (or no) degradation in terms of PDR as compared to tranceivers that are located closer to the jammer.

- **Design of a fully distributed jamming localization algorithm:** Having shown that PDR is minimized in the vicinity of the malicious device, we design a gradient descent based algorithm to locate the adversarial node. The main advantages of our approach (as compared to previously proposed localization approaches) are: (a) it is simple, (b) it does not require any special hardware support, (c) it is distributed in nature, and (d) it can be integrated with higher layer functions, such as routing, to circumvent the jammer's location.

- **Implementation and evaluation of our scheme on our testbed:** We implement our scheme on our wireless testbed using the `Click modular router` [60]. We validate its performance via experimentation; our results show that a satisfactory performance is achieved by our localization strategy; we also identify and discuss scenarios where our scheme has a difficulty in localizing the jammer.

Our work in perspective: Our goal is to exploit the inherent propagation characteristics of the wireless channel in order to expose the presence of jamming devices and localize them. The jamming attacker might be able to hide itself from all but the wireless channel's propagation characteristics. The attributes of the jamming signals (and in particular their spatial properties) can affect measurable attributes (such as the PDR) to varying degrees in different parts of the network, thereby revealing important information with regards to the location of the malicious device.

The rest of the chapter is organized as follows. Section 6.2 provides a brief description of related studies. Section 5.3 describes our analytical framework for quantifying the jamming effects on the PDR. Section 5.4 provides a full description of our algorithm. We present our experimental set-up and evaluations in Section 5.5. Section 5.6 discusses issues related to our approach. Our conclusions form Section 6.8.

5.2 Related Studies

Signal processing localization techniques: Secure mobile device localization, and in particular jammer localization, has been studied in the literature during the past years. Various approaches have been proposed in order to locate the malicious device, such as the efforts in [83, 84, 85, 86, 87]. However, all of these studies use advanced signal processing techniques and operate at the PHY layer. In addition, they require special, additional infrastructure in order to achieve their goal (e.g. ultrasound, infrared or laser infrastructures). These features make the wide deployment of such techniques rather infeasible in currently commercial wireless networks. A detailed description of various secure positioning systems, that exclusively operate at the PHY layer, can be found in [88].

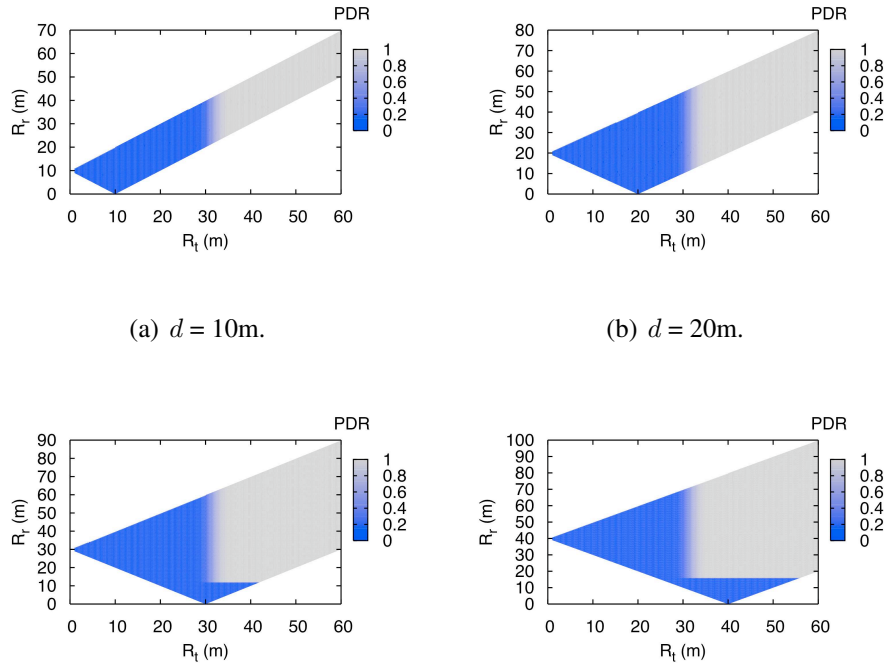
Received Signal Strength (RSS) based localization techniques: In addition to the above schemes, various studies utilize RSS measurements to discover the location of wireless devices, and in particular the positions of access points (APs). Most of these techniques require measurements of the RSS at various positions (***wardriving***). Some well known approaches belonging to this category are the *(weighted) centroid* [89] and *trilateration* [90]. Both these techniques combine

measurements of the RSS at various locations in order to infer the position of the AP. Subramanian *et al.* [91] propose a localization algorithm that utilizes steerable, directional antennas in order to get information with regards to the Angle of Arrival (AoA). This can significantly reduce the localization error. In a different approach, the authors in [92] manage to derive AoA equivalent information by simply measuring the RSS. All of these schemes, require *wardriving* and can be considered as centralized algorithms; a set of previously collected measurements, including coordinates and the corresponding RSS, are needed in order to apply the algorithms and identify the AP's position. In a slightly different context Chen *et al.* [93] combine environmental information gathered from sensor networks in order to perform localization. All the data are gathered at the base station and are analyzed in order to identify the locations needed; centralized localization is again performed.

Our approach is different from the previously proposed schemes. In particular it does not require additional, specialized infrastructure in order to operate (in contrast with signal processing systems). No changes at the driver/firmware of commercial NICs are required. Our localization system can be integrated with higher layers, as we discuss later in this chapter. One could expect that the RSS-based algorithms could be modified in order to locate a jamming node; areas close to the jamming device might exhibit extremely high RSS values due to the jamming signals [6]. However, the advantage of our approach over the RSS-based systems is that it can be executed online and in a fully distributed manner.

Gradient based routing: The idea of incorporating features from gradient optimization into network operations has been used in the past for routing. In particular, Faruque *et al.* [94] propose the use of a gradient based algorithm for the efficient forwarding of queries in sensor networks. Poor [95] presents an *on demand* routing protocol for ad hoc networks, which uses a gradient descent logic in order to forward the packets based on the *cost to destination*. In particular, the source broadcasts the message along with the cost, and only the nodes that have a smaller cost relay the packet. In a similar fashion, Ruhil *et al.* [96] forward the message to the neighbor node that is closer to the direction of the destination.

5.3 Jamming Effects on PDR



(a) d = 10m.

(b) d = 20m.

(c) d = 30m.

(d) d = 40m.

Figure 5.1: Analytical results using Equation 5.1. Shorter links are more robust, while areas around the jamming device exhibit low PDR.

The presence of a jammer can affect the PDR on a link. In particular there are 3 possible ways that a (successful) packet transmission can be affected: **(i)** the transmitter (T_x) senses the medium busy due to jamming signals, **(ii)** the reception at the receiver (R_x) fails due to low SNR at its antenna because of the jamming signals and **(iii)** the reception of the MAC layer ACK packet fails due to low SNR at the T_x antenna. Since the above events are statistically independent, the PDR can be expressed in the following way:

$$PDR = P_{T_{x_{send-DATA}}} \cdot P_{R_{x_{receive-DATA}}} \cdot P_{T_{x_{receive-ACK}}}, \tag{5.1}$$

where $P_{T_{x_{send-DATA}}}$ is the probability that T_x will sense the medium idle and transmit its packets,

$P_{R_{x_{receive-DATA}}}$ is the probability that the the SNR requirement at R_x is satisfied and $P_{T_{x_{receive-ACK}}}$ is the probability that the SNR requirement at T_x (for receiving the ACK) is satisfied too. Note that, we do not include the probability that the R_x is sensing the medium idle for the transmission of the MAC layer ACK; once R_x correctly receives the DATA packet it does not perform carrier sensing in order to send out the ACK.

In order to calculate these probabilities we need to incorporate a signal propagation model. A widely used model [9] calculates the received power P_r at distance r relative to transmission power P to be:

$$P_r = \frac{P}{r^\alpha} \cdot Y, \tag{5.2}$$

where α is the path loss exponent and Y is a random variable that is log-normally distributed. Y captures the shadow fading effects and has a mean value of 1 and a standard deviation equal to the shadow fading variation which we can obtain from measurements. Using this propagation model the components of Equation 5.1 can be expressed as follows[1]:

$$\begin{aligned} P_{T_{x_{send-DATA}}} &= P\{P_{JT} < CCA\} = Pr\{\frac{P_J}{r_T^a} \cdot Y < CCA\} \\ &= P\{Y < \frac{CCA \cdot r_T^a}{P_J}\} = \frac{1}{2} + \frac{1}{2} \cdot erf\left(\frac{ln(\frac{CCA \cdot r_T^a}{P_J}) - \mu}{\sqrt{2} \cdot \sigma}\right) \end{aligned} \tag{5.3}$$

$$\begin{aligned} P_{R_{x_{receive-DATA}}} &= P\{SNR_{R_x} > u\} = P\{Y > \frac{N \cdot u}{\frac{P}{d^a} - u \cdot \frac{P_J}{r_R^a}}\} \\ &= \frac{1}{2} - \frac{1}{2} \cdot erf\left(\frac{ln(\frac{N \cdot u}{\frac{P}{d^a} - u \cdot \frac{P_J}{r_R^a}}) - \mu}{\sqrt{2} \cdot \sigma}\right) \end{aligned} \tag{5.4}$$

[1]Note that we consider a single rate network, and in particular a network operating at the basic rate (6 Mbps).

$$P_{T_{x_{receive-ACK}}} = P\{SNR_{T_x} > u\} = P\{Y > \frac{N \cdot u}{\frac{P}{d^a} - u \cdot \frac{P_J}{r_T^a}}\}$$
$$= \frac{1}{2} - \frac{1}{2} \cdot erf \left(\frac{ln\left(\frac{N \cdot u}{\frac{P}{d^a} - u \cdot \frac{P_J}{r_T^a}}\right) - \mu}{\sqrt{2} \cdot \sigma} \right). \quad (5.5)$$

In the above equations,

- P_{JT} is the signal strength of the jamming signal at T_x,
- P_J is the transmission power of the jammer,
- r_T is the distance between the jammer and T_x,
- r_R is the distance between the jammer and the R_x,
- P is the transmission power on the link,
- d is the distance between T_x and R_x,
- u is the SNR requirement for the particular rate used (in our case 6 Mbps), and
- $(\mu,\ \sigma)$ are the parameters of the log normal distribution (computed from the mean value and the standard deviation of the r.v. Y).

Substituting Equation (5.3)-(5.5) in (5.1) provides an expression for the PDR on a link as a function of r_T and r_R. Figure 5.1 presents the PDR for various distances from the jammer and various link lengths. In generating these plots we have used the following values: (i) $P = P_J = 18$ dBm, (ii) $CCA = -80$ dBm, (iii) shadow fading signal variation is 10 dBm (value measured on our testbed) and (iv) path loss exponent is equal to 5 (this is a typical value for the path loss exponent in indoor environments [9]).

There are two main observations that we can derive from these analytical results. First, **areas in the vicinity of the jamming device (approximately 25-30m - one hop away), exhibit very low**

PDR. This forms the basis for our localization algorithm described in the following section. Second, **shorter links are more robust to jamming, since they can satisfy the SNR requirements with higher probability.**

In Section 5.5 we present experimental results for cross validation of our analysis.

5.4 Our Localization Algorithm

Gradient descent minimization: Gradient descent is a popular optimization method for real valued functions. In particular, let us assume that function f is defined on R^n and it is convex. In order to find the minimum of this function, one may start from a point $\vec{x}_0 \in R^n$ and continue finding a series of points using:

$$\vec{x}_{n+1} = \vec{x}_n - \gamma_n \cdot \nabla f(\vec{x}_n), \tag{5.6}$$

where $\nabla f(\vec{x_i})$ is the *gradient* of f. The gradient of f at point $\vec{x}$ is the direction of the maximum increase of the function at $\vec{x}$. The idea with this algorithm is that starting from a point, we greedily move towards the direction of the maximum decrease of the function at the neighborhood of this point (-$\nabla f(\vec{x}_n)$) using a step of γ_n at every iteration. After a series of iterations, the algorithm will converge to the minimum (at least a local one) of the function[2].

Our localization algorithm: As seen in the previous section, the PDR value decreases as we move to within the proximity of the jammer. Hence we can modify the above gradient descent method in order to localize the jammer. Function f is the PDR, while the next candidate points $\vec{x}_{n+1}$ are the neighbors of the node under consideration. Since we are moving in a discrete space, the PDR differential is the discrete approximation to the gradient's magnitude of the continuous function of PDR. In particular, every node will try to find its neighbor node with the largest decrease in PDR. Algorithm 3 presents a pseudocode for the algorithm executed at every node.

In the above notation, PDR_i is the PDR of node i. However, PDR is related with a link, rather

[2]Depending on the initial point, the algorithm might be trapped at a local minimum.

Data: Neighbors' PDR
Result: Next node n closer to the jammer
begin
1 Pick $k : (PDR_i - PDR_k) > (PDR_i - PDR_j)\ \forall j \neq k$
2 $\Delta = (PDR_i - PDR_k)$
3 **if** $\Delta > 0$ **then**
4 $n = k$
5 **else**
6 $n = i$
end
end
return
end

Algorithm 3: Pseudocode for the Localization Scheme for node i.

than a node. Hence, in order to calculate PDR_i we can use the average value of the PDR of the links between node i and its neighbors[3]. Specifically:

$$PDR_i = \frac{\sum_{m=1}^{|NS|} PDR_{im}}{|NS|}, \tag{5.7}$$

where NS the set of neighbors of i, PDR_{im} is the PDR on link i-m and $|NS|$ is the cardinality of set NS, i.e. the number of neighbors of node i. Using this average value makes sense since one can expect the jammer to impact the PDR on all of a victim's associated links.

5.5 System Evaluation

Testbed description: Our testbed is deployed in the 3^{rd} floor of Engineering Building 2, at the University of California, Riverside. The testbed consists of 42 nodes; 22 of them are Soekris net5501 nodes, which mount a Debian Linux distribution with kernel v2.6 over NFS and are equipped with a miniPCI *EMP-8602 6G* 802.11a/g WiFi card with Atheros chipset. The other 20 nodes are Soekris net4826, they mount the same Debian Linux distribution, and are additionally equipped with an *Intel-2915* mini-PCI card. We use 5-dBi omnidirectional antennae for every node. We use the Madwifi-ng driver for our Atheros based cards and a proprietary version of the *ipw2200*

[3]We can also pick to use the minimum or maximum value of PDR of these links.

driver/firmware of the *Intel-2915* card, which allows for tuning the CCA. More details on our testbed deployment can be found in [97].

Jammer implementation: For the purposes of our work we implement our own constant/deceptive jamming utility [6]. The implementation is based on a specific configuration (CCA = 0 dBm) and a user space utility that sends broadcast packets as fast as possible. By setting the CCA threshold to such a high value, we force the device to ignore all legitimate 802.11 signals even after carrier sensing; packets arrive at the jammer's circuitry with powers less than 0 dBm (even if the distances between the jammer and the legitimate transceivers are very small). In addition, having the jammer transmit broadcast packets allows the deferral of back-to-back transmissions for the minimum possible time[4] (i.e. $DIFS + min_{BackOff}$).

Validation of our analytical assessments: We perform experiments on our testbed in order to validate our analytical model presented in Section 5.3. We activate the jamming nodes (one at a time) and we measure the PDR observed on various links on our testbed. We perform our experiments late at night in order to avoid interference from other wireless LANs that are active during the day, and we also operate each link in isolation (no other link active at the same time).

Table 5.1 shows a subset of our experimental results in comparison with the theoretical predictions. We observe that there is a good match between the measurements and the analysis (similar matches exist for the rest of the experiments as well). However, there are some discrepancies ob-

d(m)	r_T (m)	r_R (m)	PDR measured	PDR analytical
10	32	36	0.68	0.64
10.5	18.7	18.9	0.02	0
8.1	28	25.3	0.1	0.013
7.3	30	25	0.12	0.19

Table 5.1: Our analytical model predicts well the effect of a jammer on the PDR of a link.

served that can be attributed to the fact that the path loss exponent used in the model might not match exactly with the one of the real environment. In any case (qualitative) the effects of a jammer observed on our testbed are similar with the ones expected from our analysis.

[4]Transmissions of MAC layer ACK packets are by default disabled for broadcast traffic.

System implementation details: We have implemented a prototype version of our localization scheme, using the `Click Modular Router` framework and the `Roofnet` implementation from MIT. In particular, we have modified the code at `sr2ettmetric.cpp` of the Roofnet software framework [98] in order to retrieve the (average) PDR for every node (with regards to its neighbors). Our algorithm uses these values in order to perform the localization of the jammer. The dissemination of the PDR information takes places along the lines of the ETT [99] functionality. In particular, a probe is transmitted every τ seconds and the PDR is calculated over a sliding window of w seconds (currently we have $\tau = 100ms$ and $w = 1sec$). This implementation of our algorithm allows its integration with higher layer operations (and in particular routing) with no additional overhead. In the rest of this section we present some proof-of-concept experiments on our testbed and their interpretations.

Experimental results: Our main goal is to observe how our algorithm progressively percolates through the network topology. Every node independently runs the localization algorithm and makes local decisions with regards to the next node that it is closer to the jammer, based on the PDR values of its neighbors. This procedure continues until a node cannot identify one of its neighbors as being closer to the jammer than itself. The complete percolation can be thought of as a "route discovery" propagation towards the jammer.

We illustrate the functionality of our algorithm with the following sample experiment. We activate one of our jamming devices on the testbed and run our localization algorithm on the rest nodes of our testbed in order to find the routes towards the jammer. Figure 5.2 shows the various paths towards the jammer that were reported from our algorithm for various starting points (nodes). In this experiment, the jammer is node 50 (see figure 5.2). The arrows represent the route towards the jammer that our algorithm finds, for various starting points. We extract the following interesting observations:

- Different starting points might end up into different end points.

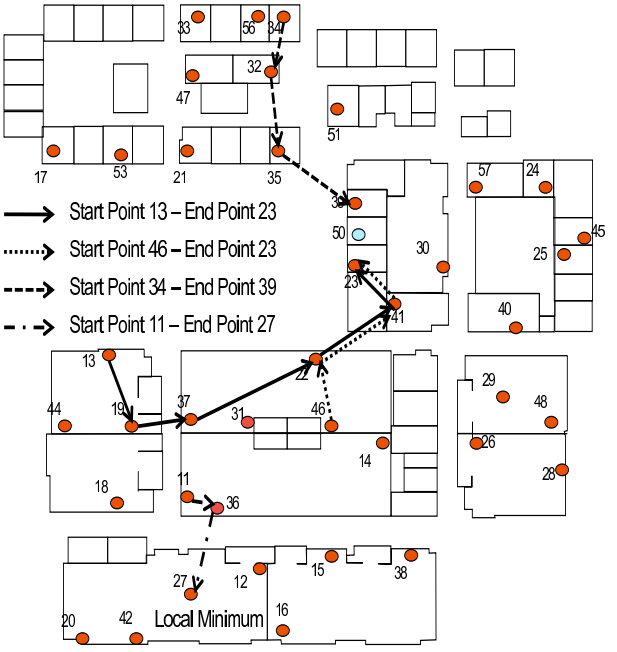


Figure 5.2: Paths to the jammer (node-50), for various starting points.

- All successful localization iterations[5] end at nodes within one hop distance from the jammer (i.e. 20-35 m).

- The paths to the jammer may be different. However, once two paths meet at an intermediate node, they *converge* and follow the same path until the termination of our algorithm.

- Depending on the starting point, our system can be trapped at a local minimum (e.g. path $11 \rightarrow 36 \rightarrow 27$). This is an inherited feature of our scheme, from gradient descent minimization technique. In Section 5.6 we discuss possible ways of overcoming this problem.

A more detailed observation at our experimental results reveals that when we start our search from nodes 13 and 46 we end up at node 23; the latter node is one hop away from the jammer. However, the paths followed are different; $13 \rightarrow 19 \rightarrow 37 \rightarrow 22 \rightarrow 41 \rightarrow 23$ and $46 \rightarrow 22 \rightarrow 41 \rightarrow 23$. Nevertheless, we have to note that once the two paths meet at node 22, they follow the same sub-route to the jammer's location. In addition, starting from node 34, we manage to

[5]With the term successful we refer to runs of our algorithm that indeed terminate *close* to the malicious device.

successfully localize the jammer once again, following a totally different path this time, that is, $34 \rightarrow 32 \rightarrow 35 \rightarrow 39$.

One side-effect from incorporating the gradient descent minimization method is that our scheme can be trapped to local minima. The performance of our proposed method is heavily dependent on the choice of the initial point/node. For the example in figure 5.2, our measurements reveal that if the localization procedure starts at node 11, it will result in a faulty localization. Specifically, our algorithm follows the path: $11 \rightarrow 36 \rightarrow 27$, and falsely concludes that the jammer is in the vicinity of node 27. This can happen for various reasons. As examples: **(a)** The links of node 27 might be inherently of bad quality (low PDR) as compared to the other links in the neighborhood of node 27 (indeed, this was the reason for being trapped to local minima in the experiment of figure 5.2). **(b)** Large-scale temporal variations in the medium can affect the performance of our localization scheme (e.g. instantaneous PDR drop due to movement of obstacles). In general, a reason for getting to local minima is the randomness of wireless channel fading. Due to channel fading randomness, it is possible that a node closer to the jammer has a higher PDR that a node further from the jammer. We elaborate on this side-effect in the following section.

5.6 Discussion and Future Directions

Sensitivity to local minima: A significant performance improvement can be attained with the elimination of the local minima sensitivity. Thus, intelligent ways that can help avoid local-minima regions are needed. One possible way could be to gather all the information with regards to PDR (collected by each legitimate node) and try to fuse these data. A majority rule could subsequently be used in order to decide upon the location of jammer(s). However, since dense-deployment regions might contain most of the nodes, the majority of the votes might still point to a local minimum. Therefore, what is required is a way to increase the confidence of the users' decisions with regards to the location of the jammer. In particular, nodes need to be able to effectively distinguish between jamming interference and heavy, legitimate interference. Xu *et al.* [6] propose

the use of *consistency checks* in order to detect the presence of a jammer. In a nutshell, if the PDR experienced is small, while the received signal strength (RSS) is high, the presence of a jammer is inferred. However, a more thorough investigation would have to be performed, since high levels of (legitimate) interference can also lead to the same observations; low PDR values in conjunction with high RSS.

Future directions: Our work reveals the potential of gradient descent based localization algorithms. Using simple metrics, such as PDR, we can localize the jammer with no additional overhead. Currently, we have implemented a prototype version in order to demonstrate the potential of this approach. In the future, we plan to examine various modifications in order to decrease the sensitivity of our algorithm to local minima, thereby improving its performance. Identifying a good starting point for our algorithm is also critical for the efficacy of localization. Furthermore, we will examine the use of different definitions for PDR_i. Using the ratio of PDR_i under jamming to the one under benign conditions or the minimum/maximum value of PDR for the links of node i, is a possible approach. This might change the number of hops required in order to reach the jammer and/or the sensitivity to local minima; we plan to further examine the applicability of such approaches. Finally, we expect that under the presence of multiple jammers our algorithm will independently reach the proximity of each; we seek to investigate the performance of our scheme under these scenarios.

5.7 Conclusions

We design a low-overhead, generic and distributed jammer localization algorithm. Our main observation that guides the construction of our system is related to the spatial effects of the jammer. In particular, links that are further from the adversary experience higher PDRs as compared to nodes that reside closer to the jamming device. We incorporate gradient descent methods in order for each network user to decide upon the immediate closer neighbor to the jammer node, from among its neighbors. The algorithm is greedy in nature; every node makes the locally optimal choice with

regard to the direction towards the jammer. Our experiments indicate that our algorithm can indeed perform efficient and effective jammer localization.

Chapter 6

ARES: An Anti-jamming REinforcement System for 802.11 Networks

Dense, unmanaged 802.11 deployments tempt saboteurs into launching jamming attacks by injecting malicious interference. Nowadays, jammers can be portable devices that transmit intermittently at low power in order to conserve energy. In this chapter, we first conduct extensive experiments on an indoor 802.11 network to assess the ability of two physical layer functions, rate adaptation and power control, in mitigating jamming. In the presence of a jammer we find that: **(a)** the use of popular rate adaptation algorithms can significantly degrade network performance and, **(b)** appropriate tuning of the carrier sensing threshold allows a transmitter to send packets even when being jammed and enables a receiver *capture* the desired signal. Based on our findings, we build ARES, an Anti-jamming REinforcement System, which tunes the parameters of rate adaptation and power control to improve the performance in the presence of jammers. ARES ensures that operations under benign conditions are unaffected. To demonstrate the effectiveness and generality of ARES, we evaluate it in three wireless testbeds: **(a)** an 802.11n WLAN with MIMO nodes, **(b)** an 802.11a/g mesh network with mobile jammers and **(c)** an 802.11a WLAN with TCP traffic. We observe that ARES improves the network throughput across all testbeds by up to 150%.

6.1 Introduction

The widespread proliferation of 802.11 wireless networks makes them an attractive target for saboteurs with jamming devices [7, 14, 15, 16]; this makes the defense against such attacks very critical. A jammer transmits electromagnetic energy to hinder legitimate communications on the wireless medium. A jamming attack can cause the following effects in an 802.11 network: **(a)** Due to carrier sensing, co-channel transmitters defer their packet transmissions for prolonged periods. **(b)** The jamming signal collides with legitimate packets at receivers. Frequency hopping techniques have been previously proposed for avoiding jammers [1] [2]. Such schemes however, are not effective in scenarios with wide-band jammers [3, 4]. Furthermore, given that 802.11 operates on relatively few frequency channels, multiple jamming devices operating on different channels can significantly hurt performance in spite of using frequency hopping [5].

In this chapter, we ask the question: *How can legacy 802.11 devices alleviate the effects of a jammer that resides on the same channel used by a legitimate communicating pair, in real time?* We address this challenge by developing ARES[1], a novel measurement driven system, which detects the presence of jammers and invokes rate adaptation and power control strategies to alleviate jamming effects. Clearly, not much can be done to mitigate jammers with unlimited resources in terms of transmission power and spectrum efficiency. Note however that in a plurality of cases the jamming device can be resource constrained, with capabilities similar to that of the legitimate device[2]. Portable, battery-operated jammers are typically configured to transmit intermittently and sometimes at low power, in order to conserve energy and harm the network for extended periods of time. In addition, misconfiguration of "legitimate" devices can transform them to resource-constrained jammers [15]. In such cases, ARES can effectively *fight* against the malicious entity, as we discuss later. Our contributions are the following:

1. Understanding the impact of jammers in an 802.11 network with rate/power control. First,

[1] ARES [pron. "áris"] was the god of war in Greek mythology; we choose the name as a symbol of the combat with jammers.

[2] We implement a jamming utility on a commodity 802.11 NIC as described in more detail in Section 6.3.

we perform an in-depth measurement-based experimental study on our indoor testbed, to quantify the impact of jamming when employing rate and/or power control. To the best of our knowledge, there are no such studies to date. With rate control, a transmitter can increase or lower its transmission rate depending on the observed packet delivery ratio (PDR) at the receiver. With power control, nodes may increase their transmission powers and/or clear channel assessment (CCA) thresholds [44] in order to increase the probability of successful packet reception. The design of ARES is driven by two key experimental observations:

i) Rate adaptation can be counter-productive: In the presence of a jammer that is active intermittently (and sleeps in between), the use of rate adaptation is not always beneficial. We conduct experiments with three popular rate adaptation algorithms: SampleRate [41], Onoe [100] and AMRR (Adaptive Multi Rate Retry) [101]. With every scheme, we observe that the use of rate adaptation may work in favor of the jammer. This is because, rate adaptation wastes a large portion of a jammer's sleeping time in order to gradually converge to the "best" rate. We analytically determine when fixed rate operations may be preferable to the use of rate adaptation.

ii) Tuning the carrier sense threshold is beneficial: We collect throughput measurements with many different transmission powers and CCA thresholds. We find that: **(a)** In the presence of a jammer, legitimate transmissions with maximum power could lead to significant benefits, only when operating at low data rates. **(b)** Increasing the CCA threshold can allow a transmitter that is being jammed to send packets and in addition, facilitate the *capture* of packets in the presence of jamming interference; together, these effects can significantly reduce the throughput degradation.

2. Designing ARES, a novel anti-jamming system. The above observations drive the design of ARES. ARES primarily consists of two modules. The ***rate control module*** chooses between fixed-rate assignment and rate adaptation, based on channel conditions and the jammer characteristics. The primary objective of this module is to effectively utilize the periods when a jammer is asleep. The ***power control module*** adjusts the CCA threshold to facilitate the transmission and the reception (*capture*) of legitimate packets during jamming. Care is taken to avoid starvation of nodes due to the creation of asymmetric links [44]. This module is used to facilitate successful com-

munications while the jammer is active. Although rate and power control have been proposed as interference alleviation techniques, their behavior has not been studied in jamming environments. To our knowledge, our work is the first to conduct such a study.

3. Implementing and experimentally validating ARES. We implement and evaluate the modules of ARES on real hardware, thereby making ARES one of the few anti-jamming system implementations for 802.11 networks. ARES relies on the existence of an accurate jamming detection module. It is beyond the scope of our work to design a new detection scheme, and thus we incorporate a mechanism proposed previously in [6]. To demonstrate the effectiveness and generality of our system, we apply it on three different experimental networks: an 802.11n WLAN with MIMO enabled nodes, an 802.11a/g mesh network with mobile jammers, and a static 802.11a WLAN with uplink TCP traffic. Our measurements demonstrate that ARES provides performance benefits in all the three networks; throughput improvements of up to 150% are observed.

6.2 Background and Related Work

In this section, first we briefly describe the operations of a jammer and its attack capabilities. Next, we discuss relevant previous studies.

Types of Jamming Attacks. Jammers can be distinguished in terms of their attack strategy; a detailed discussion can be found in [6].

Non-stop jamming: *Constant* jammers continuously emit electromagnetic energy on a channel. Nowadays, constant jammers are commercially available and easy to obtain [7, 3]. While constant jammers emit non-decipherable messages, *deceptive* jammers transmit seemingly legitimate back-to-back dummy data packets. Hence, they can mislead other nodes and monitoring systems into believing that legitimate traffic is being sent.

Intermittent Jamming: As the name suggests, these jammers are active intermittently; the primary goal is to conserve battery life. A *random* jammer typically alternates between uniformly-distributed jamming and sleeping periods; it jams for T_j seconds and then it sleeps for T_s seconds.

A *reactive* jammer starts emitting energy only if it detects traffic on the medium. This makes the jammer difficult to detect. However, implementing reactive jammers can be a challenge.

Attackers are motivated into using a random jammer because putting the jammer to sleep intermittently can increase its lifetime and decrease the probability of detection [6]. Furthermore, it is the most generalized representation of a jammer; appropriately choosing the sleep times could turn the jammer into a constant jammer or (with high probability) a reactive jammer. Moreover, reactive jammers are not easily available since they are harder to implement and require special expertise on the part of the attacker.

Related work. Most previous studies employ frequency hopping to avoid jammers. Frequency hopping, however, cannot alleviate the influence of a wide-band jammer [3, 4], which can effectively jam all the available channels. In addition, recent studies have shown that a few cleverly coordinated, narrow-band jammers can practically block the entire spectrum [5]. Thus, ARES does not rely on frequency hopping. For a set of related studies based on frequency hopping, please see [1], [2], [17].

Xu *et al.* [6] develop efficient mechanisms for jammer detection at the PHY layer (for all the 4 types of jammers). However, they do not propose any jamming mitigation mechanisms. In [8], the same authors suggest that competition strategies, where transceivers adjust their transmission powers and/or use error correction codes, *might* alleviate jamming effects. However, they neither propose an anti-jamming protocol nor perform evaluations to validate their suggestions. Lin and Noubir [77] present an analytical evaluation of the use of cryptographic interleavers with different coding mechanisms to improve the robustness of wireless LANs. In [78], the authors show that in the absence of error-correction codes (as with 802.11) the jammer can conserve battery power by destroying only a portion of a legitimate packet. Noubir [79] also proposes the use of a combination of directional antennae and node-mobility in order to alleviate jammers. ARES can easily be used in conjunction with directional antennae or with error correction codes.

6.3 Experimental Setup

In this section, we describe our wireless testbed and experimental methodology.

Testbed Description: Our testbed consists of 37 Soekris net4826 nodes [102], which mount a Debian Linux distribution with kernel v2.6, over NFS. Thirty of these nodes are each equipped with two miniPCI 802.11a/g WiFi cards, an *EMP-8602 6G* with Atheros chipset and an *Intel-2915*. The other 7 nodes are equipped with one *EMP-8602 6G* and one *RT2860* card that supports MIMO-based (802.11n) communications. We use the MadWifi driver [34] for the *EMP-8602 6G* cards. We have modified the Linux client driver [43] of the *RT2860* to enable STBC (Space Time Block Coding) support. We use a proprietary version of the *ipw2200* AP (access point) and client driver/firmware of the *Intel-2915* card. With this version we are able to tune the CCA threshold parameter.

Experimental Settings and Methodology: We experiment with different rate adaptation algorithms in the presence of random jammers. We also perform experiments with various transmission powers of jammers and powers/CCA thresholds of legitimate nodes. Our measurements encompass an exhaustive set of wireless links, routes of different lengths, as well as static and mobile jammers. We examine both SISO and MIMO links. We experiment with three modes of operation: 802.11a/g/n (unless otherwise stated throughout this chapter, our observations are consistent for all three modes of operation). The experiments are performed late at night in order to isolate the impact of the jammers by avoiding interference from co-located WLANs. By default, all devices (legitimate nodes and jammers) set their transmission powers to 18 dBm.

Implementing a random jammer: Our implementation of a jammer is based on a specific configuration (CCA = 0 dBm) and a user space utility that sends broadcast packets as fast as possible. By setting the CCA threshold to such a high value, we force the device to ignore all legitimate 802.11 signals even after carrier sensing; packets arrive at the jammer's circuitry with powers less than 0 dBm (even if the distances between the jammer and the legitimate transceivers are very small). We implement a random jammer but by setting the sleep time to zero, it can function as

a constant jammer. We use a set of 4 nodes as jammers on our testbed; these are equipped with *Intel-2915* cards which allow CCA tuning.

Traffic characteristics: We utilize the *iperf* measurement tool to generate UDP data traffic among legitimate nodes; the packet size is 1500 bytes. The duration of each experiment is 1 hour. For each experiment, we first enable *iperf* traffic between legitimate nodes, and subsequently, we activate the jammer(s). We consider both mesh and WLAN connectivity. We experiment with different jammer distributions, namely: **(a)** *frequent jammers*, which are active almost all of the time, **(b)** *rare jammers*, which spend most of their time sleeping, and **(c)** *balanced jammers* that have similar average jamming and sleeping times. We have disabled RTS/CTS message exchange throughout our experiments (a common design decision in practice [103]).

6.4 Deriving System Guidelines

In this section, we describe our experiments towards understanding the behavioral trends of power and rate adaptation techniques, in the presence of jammer(s). Our goal is to determine if there are properties that can be exploited in order to alleviate jamming effects. We perform experiments on both single-hop and multi-hop configurations.

6.4.1 Rate Adaptation in Jamming Environments

Rate adaptation algorithms are utilized to select an appropriate transmission rate as per the current channel conditions. As interference levels increase, lower data rates are dynamically chosen. Since legitimate nodes consider jammers as interferers, rate adaptation will reduce the transmission rate on legitimate links while jammers are active. Hence, one could potentially argue that rate control on legitimate links increases reliability by reducing rate and thus, can provide throughput benefits in jamming environments.

To examine the validity of this argument, we experiment with three different, popular rate adaptation algorithms, SampleRate [41], AMRR [101] and Onoe [100]. These algorithms are already

implemented on the MadWifi driver that we use. For simplicity, we first consider a balanced random jammer, which selects the sleep duration from a uniform distribution $U[1, 8]$ and the jamming duration from $U[1, 5]$ (in seconds).

Details on the experimental process: We perform experiments with both single-hop and multi-hop configurations. In each experiment, we first load the particular rate-control Linux-kernel module (SampleRate, AMRR or Onoe) on the wireless cards of legitimate nodes. We initiate data traffic between the nodes and activate the jammer after a random time. We collect throughput measurements on each data link once every 500 msec. We use the following terminology:

1) Fixed transmission rate R_f: This is the nominal transmission rate configured on the wireless card.

2) Saturated rate R_s: It is the rate achieved when R_f is chosen to be the rate on the wireless card. In order to compute R_s, for a given R_f, we consider links where the packet delivery ratio (PDR) is 100 % for the particular setting of R_f; we then measure the rate achieved in practice. We notice that for lower values of R_f, the specified rate is actually achieved on such links. However, for higher values of R_f (as an example $R_f = 54$ Mbps), the achieved data rate is much lower; this has been observed in other work e.g. [104]. Table 6.1 contains a mapping, derived from measurements on our testbed, between R_f and R_s.

3) Application data rate R_a: This is the rate at which the application generates data.

R_f	6	9	12	18	24	36	48	54
R_s	6	9	12	18	24	26	27	27

Table 6.1: The saturated-throughput matrix in Mbps.

It is difficult (if not impossible) to a priori determine the *best* fixed rate on a link. Given this, we set:

$$R_f = \{min\ R_f :\ R_f \geq R_a\},$$

which is the maximum rate that is required by the application (we discuss the implications of this choice later). Our key observations are summarized below:

- **Rate adaptation algorithms perform poorly on high-quality links due to the long times**

that they incur for converging to the appropriate high rate.

- **On *lossless* links, the fixed rate R_f is better, while rate adaptation is beneficial on *lossy* links.**

We defer defining what constitute lossless or lossy links to later; conceptually, we consider lossless links to be those links that can achieve higher long-term throughput using a fixed transmission rate R_f , rather than by applying rate adaptation.

Single-hop Configurations

Our experiments with one-hop connectivity involve 80 sets of sender-receiver pairs and one jammer per pair. We impose that a jammer interferes with *one* link at a time and that the legitimate data links do not interfere with each other. Thus, we perform 20 different sets of experiments, with 4 isolated data links and 4 jammers in each experiment.

Rate adaptation consumes a significant part of the jammer's sleep time, to converge to the appropriate rate: As soon as the jammer "goes to sleep", the link quality improves and thus, the rate control algorithm starts increasing the rate progressively. However, since the purpose of a jamming attack is to corrupt as many transmissions as possible, the jammer will typically not sleep for a long time. In such a case, the sleep duration of the jammer will not be enough for the rate control to reach the highest rate possible. To illustrate this we choose two links on our testbed, one that can support 12 Mbps and the other that can support 54 Mbps. Figure 6.1 depicts the results. We observe that **(a)** irrespective of whether SampleRate or a fixed rate strategy is used, during jamming the throughput drops to values close to zero since the jammer blocks the medium for the sender, and **(b)** *the throughput achieved with SampleRate is quite low, and much lower than if we fix the rate to the constant value of 12 Mbps.* Note that we have observed the same behavior with AMRR and Onoe.

Fixed rate assignment outperforms rate adaptation on lossless links: As alluded to above, in order to find the *best* rate on a link after a period where there is no throughput due to a jammer, the

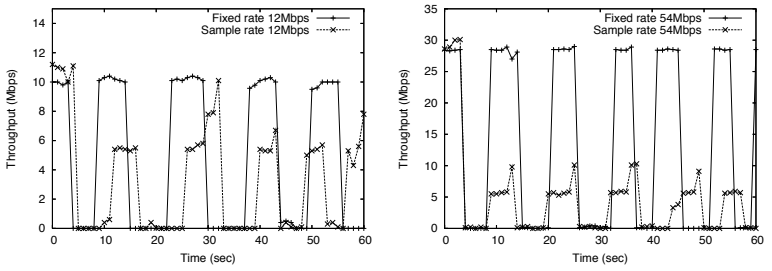


Figure 6.1: Rate adaptation algorithms may not find the best rate during the sleep period of the jammer. We show cases for $R_a = 12$ Mbps (left) and $R_a = 54$ Mbps (right).

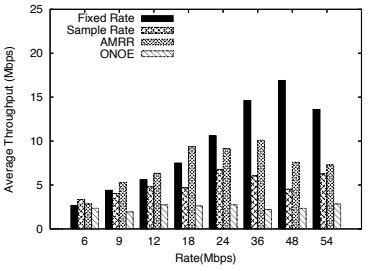


Figure 6.2: Fixed rates outperform rate adaptation for high-quality links, under random jamming. ($R_a = R_f$)

rate adaptation mechanisms gradually increase the rate, invoking transmissions at all the lower rates interim, until the best rate is reached. For links that can inherently support high rates, this process might consume the sleep period of the jammer (as suggested by the results in Figure 6.1). If the best rate for a link was known a priori, at the instance that the jammer goes to sleep, transmissions may be invoked at that rate. This would utilize the sleep period of the jammer more effectively. As observed in Figure 6.2, the throughputs achieved with fixed rate assignment are much higher than those achieved with rate adaptation on such links.

Determining the right transmission rate policy:

Implications of setting $R_f = \{min\ R_f :\ R_f \geq R_a\}$*:* Since the application does not require the link to sustain a higher rate, the highest throughput for that application rate is reached either with this choice of R_f or with some rate that is lower than R_a. If the rate adaptation algorithm converges to a rate that results in a throughput that is higher than with the chosen R_f, then the adaptive rate strategy should be used. If instead, during the jammer's sleep period, the rate adaptation technique

is unable to converge to such a rate, the fixed rate strategy is better.

Analytically determining the right rate: In order to determine whether it is better to use a fixed or an adaptive-rate approach for a given link, we perform an analysis based on the following parameters:

1. The distribution of the jammer's active and sleep periods (we call this the *jammer's distribution*).

2. The application data rate, R_a.

3. The performance metric on the considered legitimate link, i.e., PDR, link throughput, etc.

4. The rate adaptation scheme that is employed, i.e., Onoe, SampleRate, etc. The key scheme-specific factor is the transition time from a lower rate to the next higher rate, under conducive conditions.

5. The *effectiveness* of the jammer F, measured by the achievable throughput while the jammer is on. The lower the throughput, the more effective the jammer.

Let us suppose that the expected *sleeping* duration of the jammer during a cycle, is given by $E[t_s]$ and the expected period for which it is active, by $E[t_j]$. The expected duration of a *cycle* is then $E[t_s] + E[t_j]$. As an example, if the jammer picks its sleeping period from a uniform distribution $U[a, b]$ and its jamming period from $U[c, d]$, $E[t_s]$ and $E[t_j]$ are equal to $\frac{b+a}{2}$ and $\frac{d+c}{2}$, respectively. For simplicity let us assume that the link-quality metric employed[3] is the PDR. With application data rate R_a and ***fixed*** transmission rate R_f, the throughput achieved during a jammer's cycle is:

$$T_{fixed} = \frac{E[t_s]}{E[t_s] + E[t_j]} \cdot PDR_f \cdot R_s + \frac{E[t_j]}{E[t_s] + E[t_j]} \cdot F, \tag{6.1}$$

where PDR_f is the PDR of the link at rate R_f. Recall that the rate achieved in practice with a specified rate R_f is R_s.

[3]Our analysis can be modified to adopt any other link-quality metric.

To compute the throughput with ***rate adaptation***, we proceed as follows. Let us assume that $x(F, R_s)$ corresponds to the convergence time of the rate adaptation algorithm (specific to the chosen algorithm). We consider the following two cases.

1) $\boldsymbol{x(F, R_s) < E[t_s]}$. This case holds when the jammer's sleep duration is sufficient (on average) for the rate control algorithm to converge to the best rate R_s. In this scenario, the achievable throughput is:

$$T_{adapt} = \frac{[E[t_s] - x(R_s)]\cdot R_s + \sum_{R_i} y(R_i)\cdot R_i + E[t_j]\cdot F}{E[t_s] + E[t_j]},$$

where $R_i \in S$, S being the set of all intermediate rates from F to R_s. $y(R_i)$ is the time that the rate control algorithm spends at the corresponding rate R_i. The values of $y(R_i)$ are specific to the implementation of the rate control algorithm. Note that $x(F, R_s)$ can be easily computed from $y(R_i)$ by adding all the individual durations for the rates belonging to the set S.

2) $\boldsymbol{x(F, R_s) \geq E[t_s]}$. In this scenario, the average sleep time of the jammer is insufficient for the rate control algorithm to converge to the desired rate. When the jammer wakes up, the rate will again drop due to increased interference. Here, the throughput that can be achieved during a jammer's cycle is:

$$T_{adapt} = \frac{\sum_{i=1}^{n} y(R_i)\cdot R_i + \left[E[t_s] - \sum_{i=1}^{n} y(R_i)\right]\cdot R_{n+1} + E[t_j]\cdot F}{E[t_s] + E[t_j]}$$

where $n = max\{k : \sum_{i=1}^{k} y(R_i) \leq E[t_s]\ \}$.

Based on the above analysis, we define a link to be **lossy**, when $T_{fixed} \leq T_{adapt}$; the links on which $T_{fixed} > T_{adapt}$ are classified as **lossless** links. Clearly for lossy links it is better to use the rate adaptation algorithm. The analysis can be used to compute PDR_f^{TH}, a threshold value of PDR_f below which, a rate adaptation strategy performs better than the fixed rate approach. In particular, by setting $T_{fixed} = T_{adapt}$ and solving this equation, one can compute PDR_f^{TH}. Based on this, a decision can be made on whether to enable rate adaptation or use fixed-rate assignment.

If the observed PDR is larger than the computed threshold, fixed rate should be used; otherwise, rate adaptation should be used.

Validation of our analysis: In order to validate our analysis, we measure PDR_f^{TH} on 80 different links in the presence of a balanced jammer. We then compare them against the PDR_f^{TH} values computed with our analysis. Note here that the analysis itself depends on measured values of certain quantities (such as the jammer distribution and the function $y(R_i)$). In this experiment, we consider the SampleRate algorithm, and measure the values of $x(F, R_s)$ and $y(R_i)$. The jammer's sleep time follows $U[0, 4]$ and the jamming time follows $U[1, 6]$. Figure 6.3 plots the values of function y for different values of R_f.

In Table 6.2, we compare the theoretically computed PDR thresholds with the ones measured on our testbed, for various values of R_f. We observe that the PDR_f thresholds computed with our analysis are very similar to the ones measured on our testbed. There are slight discrepancies since our analysis is based on using measured average values which may change to some extent over time. We wish to stress that while we verify our analysis assuming that the jammer is active and idle for uniformly distributed periods of time, our analysis depends only on expected values and is therefore valid for other jammer distributions. Finally, Figure 6.4 shows the advantage of using a fixed rate approach over SampleRate for various PDR values and with R_f = 54 Mbps. We observe that SampleRate provides higher throughput only for very low PDR values.

R_f	Measured PDR_f^{TH}	Analytical PDR_f^{TH}
6	0.82	0.83
9	0.52	0.55
12	0.40	0.41
18	0.26	0.27
24	0.19	0.21
36	0.19	0.20
48	0.17	0.185
54	0.15	0.185

Table 6.2: PDR_f thresholds

Next, we consider two extreme cases of jamming: frequent and rare jammers (see section 6.3).

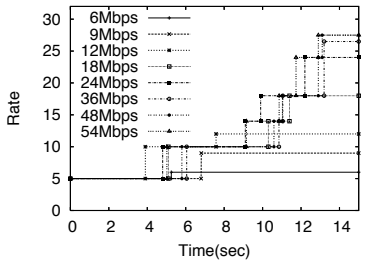


Figure 6.3: Measured convergence times of the MadWifi SampleRate algorithm, for the different application data rates.

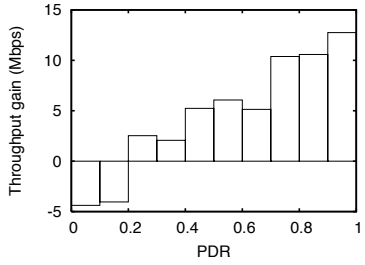


Figure 6.4: Throughput gain of fixed rate Vs. SampleRate, for various link qualities and for application data rate of 54 Mbps.

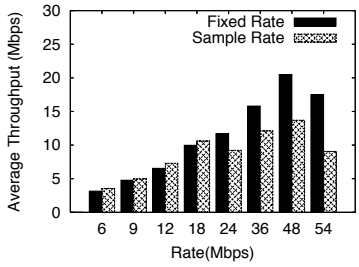


Figure 6.5: The performance with rare jammers is aligned with our observations for the case with balanced jammers. ($R_a = R_f$)

The distributions that we use in our experiments for these jammers are shown in Table 6.3. Note that by choosing the jammer's sleeping and jamming time from distributions like that of the frequent jammer, we essentially construct a constant jammer. With frequent jammers, the difference in the performance between fixed rate assignment and rate adaptation is larger, while for a rare jammer it is smaller. This is because with rare jamming, rate adaptation has more time to converge and therefore often succeeds in achieving the highest rate possible; one observes the opposite effect when we have a frequent jammer. The results are plotted in Figures 6.5 and 6.6.

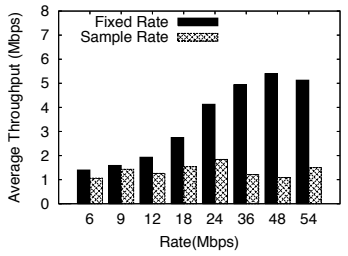


Figure 6.6: Fixed rate improves the performance more than rate adaptation at high rates, with frequent jammers. ($R_a = R_f$)

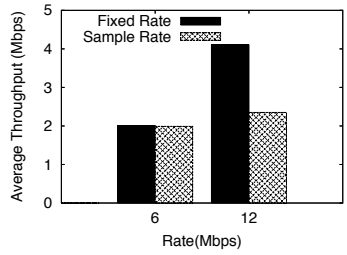


Figure 6.7: Rate adaptation presents the same behavior in multihop links; it provides lower throughput at high rates.

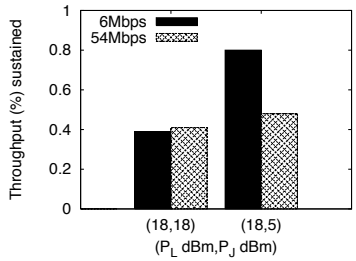


Figure 6.8: Percentage of the isolated throughput, for various P_L and P_J combinations, for two different transmission rates.

-	Sleep time (sec)	Jamming time (sec)
Balanced	U[1,8]	U[1,5]
Rare	U[1,5]	U[1,2]
Frequent	U[1,2]	U[1,15]

Table 6.3: The jamming distributions that we use in our experiments.

Random Jamming in Multi-hop Topologies

Next, we examine the impact of a random jammer on the end-to-end throughput of a multi-hop path. We experiment with 15 different routes on our testbed. We fix static routes of various lengths (from 2 to 4 links per route) utilizing the *route* Unix tool in order to modify the routing tables of nodes. We place a jammer such that it affects one or more links. Along each route, links that are not affected by the jammer consistently use a rate adaptation algorithm. *On the links that are subject to jamming, our analysis dictates the decision on whether to use fixed or adaptive rate assignment.*

We measure the end-to-end throughput on the route. We show our results for routes on which, in the absence of a jammer, end-to-end throughput of 6 and 12 Mbps was observed. From Figure 6.7 we see that *the behavior with rate adaptation on multi-hop routes, in the presence of a random jammer, is the same as that on a single-hop link*. In particular, with low data rates, a sufficiently high PDR has to be sustained over the route, in order for a fixed rate approach to perform better than rate adaptation. On the other hand, when routes support high data rates, fixing the rate on the individual links (that are affected by the jammer) as per our analytical framework, provides higher benefits.

Choosing the right policy in practice: To summarize our findings, our analysis demonstrates that using a fixed rate may be attractive on lossless links while it would be better to use rate adaptation on lossy links. However, as discussed, determining when to use one over the other in real time during system operations is difficult; the determination requires the knowledge of $x(F, R_s)$, $y(R_i)$ and estimates of how often the jammer is active/asleep, on average. Thus, we choose a simpler practical approach that we call MRC for Markovian Rate Control. We will describe MRC in detail later (in section 6.5) but in a nutshell, MRC induces memory into the system and keeps track of the feasible rates during benign jamming-free periods; as soon as the jammer goes to sleep, legitimate transmissions are invoked at the most recent rate used during the previous sleeping cycle of the jammer. We also perform offline measurements by directly using our analytical formulation (with knowledge of the aforementioned parameters); these measurements serve as benchmarks for evaluating the efficacy of MRC (discussed in section 6.6).

6.4.2 Performance of Power Control in the Presence of Random Jamming

Next, we examine whether tuning power levels can help cope with the interference injected by a jammer. If we consider a single legitimate data link and a jammer, incrementing the transmission power on the data link should increase the SINR (signal-to-interference plus noise ratio) of the received data packets. Thus, one could argue that increasing the transmission power is always beneficial in jamming environments [77].

We vary the transmission powers of both the jammer and legitimate transceiver, as well as the CCA threshold of the latter. Note that the jammer's transmission distribution is not very relevant in this part of our study. Our expectation is that tuning the power of legitimate transceivers will provide benefits while the jammer is active. ***In other words, one can expect that the benefits from power control will be similar with any type of jammer.*** We define the following:

- $\boldsymbol{RSSI_{TR}}$: The RSSI of the signal of the legitimate transmitter at its receiver.
- $\boldsymbol{RSSI_{RT}}$: The RSSI of the signal in the reverse direction (the receiver is now the transmitter).
- $\boldsymbol{RSSI_{JT}}$ and $\boldsymbol{RSSI_{JR}}$: The RSSI values of the jamming signal at the legitimate transmitter and receiver, respectively.
- $\boldsymbol{RSSI_J}$: The minimum of $\{RSSI_{JT},\ RSSI_{JR}\}$.
- $\boldsymbol{P_L}$ and $\boldsymbol{CCA_L}$: The transmission power and the CCA threshold at legitimate transceivers.
- $\boldsymbol{P_J}$: The transmission power of the jammer.

Our main observations are the following:

- **Mitigating jamming effects by incrementing P_L is viable at low data rates. It is extremely difficult to overcome the jamming interference at high rates, *simply with power adaptation*.**
- **Increasing CCA_L restores (in most cases) the isolated throughput (the throughput achieved in the absence of jammers).**

We present our experiments and the interpretations thereof, in what follows.

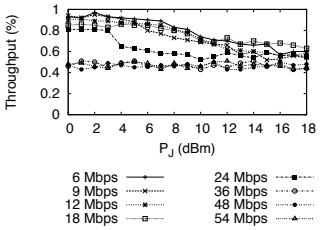


Figure 6.9: Percentage of the isolated throughput in the presence of a balanced jammer for various P_J and P_J values and data rates.

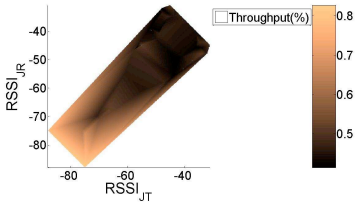


Figure 6.10: Percentage of the isolated throughput in the presence of a balanced jammer Vs. $RSSI_J$, for CCA_L= –80 dBm.

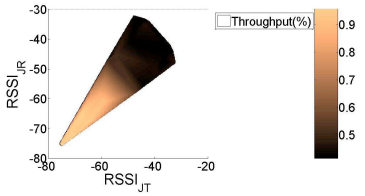


Figure 6.11: Percentage of the isolated throughput, for various $RSSI_J$ values, and for CCA_L = –50 dBm.

Increasing P_L to cope with jamming interference

Increasing P_L will increase the SINR and one might expect that this would reduce the impact of jamming interference on the throughput. In our experiments we quantify the gains from employing such a "brute-force" approach.

Details on the experimental process: We perform measurements on 80 different links and with 4 jammers. We consider different fixed values for P_J (from 1 dBm to 18 dBm). For each of these values we vary P_L between 1 and 18 dBm and observe the throughput in the presence of the jammer, for all possible fixed transmission rates. For each chosen pair of values $\{P_L, P_J\}$, we run 60-minute repeated experiments and collect a new throughput measurement once every 0.5 seconds. Both end-nodes of a legitimate link use the same transmission power.

The combination of high P_L and low data rate helps mitigate the impact of low-power jammers. We experiment with many different locations of the jammers. Our measurements indi-

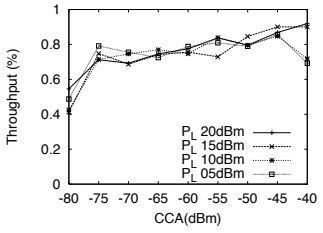


Figure 6.12: Percentage of the isolated throughput, for various CCA_L values and various P_L values. P_J = 20 dBm.

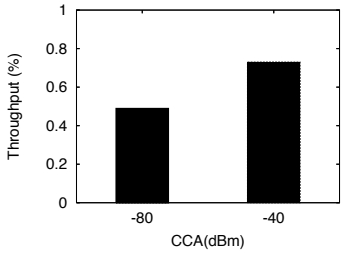


Figure 6.13: Careful CCA adaptation significantly improves the end-to-end throughput along a route.

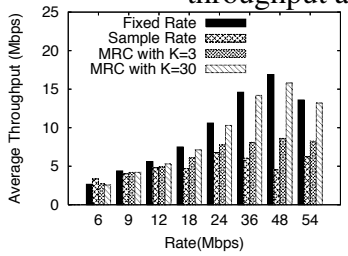


Figure 6.14: MRC outperforms current rate adaptation algorithms, especially for high values of K.

cate that when high transmission rates are used, increasing P_L does not help alleviate the impact of jammers. Sample results are depicted in Figure 6.8. In this figure, we plot the percentage of the isolated throughput achieved in the presence of jamming, for two representative combinations of P_L and P_J and for 2 different rates. In our experiments on the 80 considered links, *there were no links where incrementing P_L increased the throughput at high data rates, even with very low jamming powers.* While there could exist cases where incrementing P_L could yield benefits at high rates, this was not observed. In contrast, we observe that with low data rates and when P_J is low, data links can overcome jamming to a large extent by increasing P_L. Figure 6.9 depicts another representative subset of our measurement results where all legitimate nodes use P_L=18 dBm, while P_J is varied between 1 and 18 dBm. We observe that the combination of high P_L with low data rate helps overcome the impact of jamming, when P_J is low. Note also that when P_J is high, it is extremely difficult to achieve high average throughput.

The above observations can be explained by taking a careful look at the following two cases:

Strong jammer: Let us consider a jammer such that $RSSI_J > CCA_L$. This can result in two effects: **(a)** The sender will sense that the medium is constantly busy and will defer its packet transmissions for prolonged periods of time. **(b)** The signals of both the sender and the jammer will arrive at the receiver with RSSI values higher than CCA_L. This will result in a packet collision at the receiver. In both cases, the throughput is degraded. Our measurements show that *it is not possible to mitigate strong jammers simply by increasing* P_L.

Weak jammer: Let us suppose that the jammer's signals arrive with low RSSI at legitimate nodes. This may be either due to energy-conservation strategies implemented by the jammer causing it to use low P_J (e.g., 2 dBm), or due to poor channel conditions between a jammer and a legitimate transceiver. At high transmission rates, the SINR required for the successful decoding of a packet is larger than what is required at low rates (shown in Table 6.4) [44]. Our throughput measurements show that even in the presence of weak jammers, the SINR requirements at high transmission rates are typically not satisfied. However, since the SINR requirements at lower data rates are less stringent, *the combination of high* P_L *and low rate, provides significant throughput benefits*.

Data Rate	6	9	12	18	24	36	48	54
SINR (dB)	6	7.8	9	10.8	17	18.8	24	24.6

Table 6.4: SINR levels required for successful packet decoding, in 802.11a/g.

Tuning CCA_L on single-hop settings

Next, we investigate the potential of adjusting CCA_L in conjunction with P_L.

Implementation and experimental details: For these experiments we exclusively use the *Intel-2915* cards; these cards allow us to tune the CCA threshold. We have modified a prototype version of the AP/client driver, in order to periodically collect measurements for $RSSI_{TR}$, $RSSI_{RT}$ and $RSSI_J$. We consider 80 AP-client data links, with traffic flowing from the AP to the client. As before, we divide the 80 data links into 20 sets of 4 isolated links. We use Intel's proprietary rate

adaptation algorithm, which has been implemented in the firmware of the *Intel-2915* cards. We measure the achieved data throughput for different values of P_L and CCA_L. Both nodes of a data link use the same power and CCA threshold values.

Tuning the CCA threshold is a potential jamming mitigation technique. To begin with, we perform throughput measurements with the default CCA_L value (-80 dBm), and with various $RSSI_J$ values. We observe from Figure 6.10 that when $RSSI_J < CCA_L$, data links achieve high throughput. This is because signals with RSSI $< CCA_L$ are ignored by the transceiver's hardware. In particular, **(a)** such signals do not render the medium busy, and **(b)** receivers are trying to latch onto signals with RSSI $> CCA_L$, while other signals are considered to be background noise. Moreover, even when $RSSI_J$ is slightly larger than CCA_L, we still observe decent throughput achievements for the cases wherein data links operate at high SINR regimes. This is because the reported RSSI value is an average and the jammer signal could be below the threshold even here, in many cases. These measurements imply that the ability to tune CCA_L can help receive data packets correctly, even while jammers are active.

In order to further explore the potential of such an approach, we vary CCA_L from -75 to -30 dBm on each of the considered 80 links. Figure 6.11 depicts the results for the case where CCA_L is equal to -50 dBm.We observe that ***increasing*** CCA_L ***results in significantly higher data throughput, even with quite high*** $RSSI_J$ ***values.*** More specifically, from Figure 6.11 we observe that when $RSSI_J$ is lower than CCA_L, links can achieve up to 95% of the throughput that is achieved when the medium is jamming free. When $RSSI_J \approx CCA_L$, data links still achieve up to 70% of the jamming-free throughput (capture of data packets is still possible to a significant extent). As one might expect, if $RSSI_J \gg CCA_L$, there are no performance benefits.

Our observations also hold in some scenarios where, $P_J > P_L$. Figure 6.12 presents the results from one such scenario. We observe that appropriate CCA settings can allow legitimate nodes to exchange traffic effectively, even when $P_J \gg P_L$. This is possible if the link conditions between the jammer and the legitimate transceivers are poor and result in low $RSSI_J$. Note here that one cannot increase CCA_L to arbitrarily high values on legitimate nodes. Doing so is likely to compromise

connectivity between nodes or degrade the throughput due to failure of capturing packets as seen in Figure 6.12 for $P_L = 5dBm$ and $P_L = 10dBm$.

Tuning CCA_L in multi-hop configurations

We perform experiments with various CCA thresholds along a route. Previous studies have shown that in order to avoid starvation due to asymmetric links, the transmission power and the CCA threshold need to be jointly tuned for all nodes of the same connected (sub)network [44]. In particular, the product $C = P_L \cdot CCA_L$ must be the same for all nodes. Given this, we ensure that C is the same for all nodes that are part of a route. In particular, we set P_L to be equal to the maximum possible value of 20 dBm on all nodes of a route; for each run, CCA_L is therefore set to be the same on all of the nodes on the route. Throughout our experiments with multi-hop traffic, nodes on one route do not interfere with nodes that are on other routes. In scenarios where nodes belonging to different routes interfere with each other, if all nodes use the same P_L, their CCA_L values must be the same [44], [45]. However, we did not experiment with such scenarios given that our objective is to isolate the impact of a jammer and not to examine interference between coexisting sessions in a network.

We experiment with the same multi-hop settings as in section 6.4.1. Figure 6.13 presents the results observed on one of our routes. We observe that careful CCA tuning can provide significant average end-to-end throughput benefits along a route.

6.5 Designing ARES

In this section, we design our system ARES based on the observations from the previous section. ARES is composed of two main modules: **(a)** a *rate module* that chooses between fixed or adaptive-rate assignment, and **(b)** a *power control module* that facilitates appropriate CCA tuning on legitimate nodes.

Rate Module in ARES: As discussed in section 6.4.1, our experiments with three popular rate adaptation algorithms show that the convergence time of the algorithms affects the link performance in random-jamming environments. This convergence time is largely implementation specific. As an example, our experiments with both SampleRate and Onoe show that in many cases it takes more than 10 sec for both algorithms to converge to the "best" rate; [105] reports similar observations. The rate module in ARES decides on whether a fixed or an adaptive-rate approach should be applied.

MRC: Markovian Rate Control: MRC is an algorithm–patch that can be implemented on top of any rate control algorithm. MRC is motivated by our analysis in section 6.4. However, as discussed earlier, it does not directly apply the analysis, since this would require extensive offline measurements (the collection of which can be time-consuming) and estimates of the jammer active and sleep periods. The key idea that drives MRC is that a rate adaptation algorithm need not examine the performance at all the transmission rates during the sleeping period of the jammer. The algorithm simply needs to remember the previously used transmission rate, and use it as soon as the jammer goes to sleep. Simply put, MRC introduces *memory* into the system. The system keeps track of past transmission rates and hops to the stored highest-rate state as soon as the jammer goes to sleep. Since the channel conditions may also change due to the variability in the environment, MRC invokes the re-scanning of all rates periodically, once every K consecutive sleeping/jamming cycles. When $K = 1$ we do not expect to have any benefits, since the scanning takes place in each cycle.

Note here that the appropriate value of K depends on the environment and the sleep and active periods of the jammer. One could adaptively tune the K value. As an example, an additive increase additive decrease strategy may be used where one would increase the value of K until a degradation is seen. The K value would then be decreased. The implementation of such a strategy is beyond the scope of this thesis and will be considered in the future.

Implementation details of MRC: The implementation **(a)** keeps track of the highest transmission rate used over a benign time period (when the jammer is asleep) and, **(b)** applies this rate imme-

diately upon the detection of the next transition from the jammer's active period to the sleeping period.

Figure 6.14 presents a set of measurements with MRC, with intermittent SampleRate invocations (once every K cycles) for $K = \{3, 30\}$. We observe that MRC outperforms pure SampleRate in jamming environments, especially with larger values of K. With small K, the rate adaptation algorithm is invoked often and this reduces the achieved benefits. Furthermore, MRC provides throughput that is close to the maximum achievable on the link (which may be either with fixed or adaptive rate, depending on whether the link is lossy or lossless).

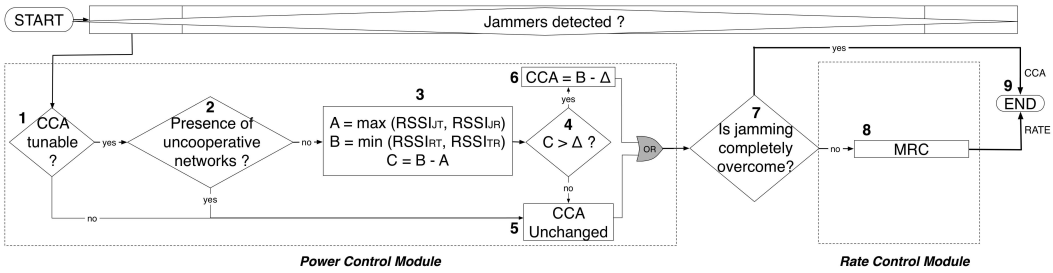


Figure 6.15: ARES: our Anti-jamming Reinforcement System.

Power Control Module in ARES: As discussed in section 6.4.2, increasing P_L is beneficial at low rates; while at high rates this is not particularly useful, it does not hurt either. Since our goal in this chapter is to propose methods for overcoming the effects of jamming (and not legitimate) interference, we impose the use of the maximum P_L by all nodes in the presence of jammers. The design of a power control mechanism that in addition takes into account the imposed legitimate interference (due to high P_L) is beyond the scope of this thesis.

More significantly, our power control module overcomes jamming interference by adaptively tuning CCA_L. The module requires the following inputs on each link:

- The values of $RSSI_{TR}$, $RSSI_{RT}$, $RSSI_{JR}$, and $RSSI_{JT}$. These values can be easily observed in real time.

- An estimation for the shadow fading variation of the channel, Δ. Due to shadow fading,

the above RSSI values can occasionally vary by Δ. The value of Δ is dependent on the environment of deployment. One can perform offline measurements and configure the value of Δ in ARES.

We determine the variations in RSSI measurements via experiments on a large set of links. The measurements indicate that Δ is approximately 5 dB for our testbed (a less conservative value than what is reported in [61]). The value of CCA_L has to be at least Δ dB lower than both $RSSI_{TR}$ and $RSSI_{RT}$, to guarantee connectivity at all times. Hence, ARES sets:

$$CCA_L = min(RSSI_{TR}, RSSI_{RT}) - \Delta, \quad \text{if}$$

$$max(RSSI_{JT}, RSSI_{JR}) \leq min(RSSI_{TR}, RSSI_{RT}) - \Delta.$$

Otherwise, CCA_L is not changed[4]. This ensures that legitimate nodes are always connected, while the jammer's signal is ignored to the extent possible. Our experiments indicate that, especially if

$$max(RSSI_{JT}, RSSI_{JR}) \leq min(RSSI_{TR}, RSSI_{RT}) - 2\Delta,$$

the data link can operate as if it is jamming-free.

In order to avoid starvation effects, the tuning of the CCA threshold should be performed only when nodes that participate in power control belong to the same network [45]. Unless collocated networks cooperate in jointly tuning their CCA (as per our scheme), our power control module will not be used. Note that when jamming attacks become more prevalent, cooperation between coexisting networks may be essential in order to fight the attackers. Hence, in such cases collocated networks can have an agreement to jointly increase the CCA thresholds when there is a jammer.

Implementation details: Our power control algorithm can be applied in a **centralized** manner by having all legitimate nodes report the required RSSI values to a central server. The central server then applies the same CCA_L value to all nodes (of the same connected network). The cho-

[4]We choose not to tune CCA_L, unless we are certain that it can help alleviate jamming interference.

sen CCA_L is the highest possible CCA threshold that guarantees connectivity between legitimate nodes. This reporting requires trivial modifications on the wireless drivers. We have implemented a centralized functionality when our network is configured as a multi-hop wireless mesh.

In a **distributed** setting, our algorithm is applicable as long as legitimate nodes are able to exchange RSSI information. Each node can then independently determine the CCA_L value. To demonstrate its viability, we implement and test a distributed version of the power control module in a 802.11a/g WLAN configuration. In particular, we modify the Intel prototype AP driver, by adding an extra field in the "Beacon" template. This new field contains a matrix of RSSI values of neighboring jammers and legitimate nodes. We enable the decoding of received beacons in the AP driver (they do not read these by default). Assuming that a jammer imposes almost the same amount of interference on all devices (AP and clients) within a cell, the AP of the cell determines the final CCA_L after a series of iterations in a manner very similar to the approaches in [45], [44].

Combining the modules to form ARES: We combine our rate and power control modules to construct ARES as shown in Figure 6.15. The goal of ARES is to apply the individual modules as appropriate, once the jammers are detected. For the latter, ARES relies on already existing jamming detection schemes and inherits their accuracy. For example, the mechanism that was proposed in [6] can be used; this functionality performs a consistency check between the instantaneous PDR and RSSI values. If the PDR is extremely low while the RSSI is much higher than the default CCA_L, the node is considered to be jammed. We want to reiterate, that it is beyond the scope of our work to design a new, even more accurate, detection scheme.

ARES applies the power control module first, since with this module, the impact of the jammer(s) could be completely overcome. If the receiver is able to capture and decode all packets in spite of the jammer's transmissions, no further actions are required. Note that even if $CCA_L > RSSI_J$, the jammer can still affect the link performance. This is because with CCA tuning the jamming signal's power is added to the noise power. Hence, even though the throughput may increase, the link may not achieve the "jamming-free performance" while the jammer is active. If the jammer still has an effect on the network performance after tuning CCA_L, (or if CCA

tuning is infeasible due to the presence of collocated uncooperative networks) ARES enables the rate module. Note that the two modules can operate independently and the system can bypass any of them in case the hardware/software does not support the specific functionality.

6.6 Evaluating our system

We first evaluate ARES by examining its performance in three different networks: a MIMO-based WLAN, an 802.11 mesh network in the presence of mobile-jammers, and an 802.11a WLAN setting where uplink TCP traffic is considered.

ARES boosts the throughput of our MIMO WLAN under jamming by as much as 100%: Our objective here is twofold. First, we seek to observe and understand the behavior of MIMO networks in the presence of jamming. Second, we wish to measure the effectiveness of ARES in such settings. Towards this, we deploy a set of 7 nodes equipped with *Ralink RT2860* miniPCI cards.

Experimental set-up: We examine the case for a WLAN setting, since the *RT2860* driver does not currently support the ad-hoc mode of operations. MIMO links with Space-Time Block Codes (STBC) are expected to provide robustness to signal variations, thereby reducing the average SINR that is required for achieving a desired bit error rate, as compared to a corresponding SISO (Single-Input Single-output) link. For our experiments, we consider 2 APs, with 2 and 3 clients each, and two jammers. Fully-saturated downlink UDP traffic flows from each AP to its clients.

Applying ARES on a MIMO-based WLAN: We first run experiments without enabling ARES. Interestingly, we observe that in spite of the fact that STBC is used, 802.11n links present the same vulnerabilities as 802.11a or g links. In other words, MIMO does not offer significant benefits by itself, in the presence of a jammer. This is due to the fact that 802.11n is still employing CSMA/CA and as a result the jamming signals can render the medium busy for a MIMO node as well. Moreover, for STBC codes to work effectively and provide a reduction in the SINR for a desired bit error rate (BER), the signals received on the two antenna elements will have to experience independent

multipath fading effects. In other words, a line of sight or dominant path must be absent. However, in our indoor testbed, given the proximity of the communicating transceiver pair, this may not be the case. Thus, little diversity is achieved [42] and does not suffice in coping with the jamming effects.

Next, we apply ARES and observe the behavior. The logical set of steps that ARES follows (in Figure 6.15) is $1 \rightarrow 5 \rightarrow 7 \rightarrow 8 \rightarrow 9$. Since the CCA threshold is not tunable with the *RT2860* cards, ARES derives decisions with regards to rate control only. Figure 6.16 depicts the results. We observe that the configuration with ARES outperforms the rate adaptation scheme that is implemented on the *RT2860* cards in the presence of the jammer, by as much as 100%. Note that higher gains would be possible, if ARES was able to invoke the power control module.

In Figure 6.16 we also compare the throughput with MRC against the suggested settings with our analysis (these settings allow us to obtain benchmark measurements possible with global information). The parameters input to the analysis are the following: **(a)** The jammer is balanced with a jamming distribution $U[1, 5]$ and a sleep distribution $U[1, 6]$. **(b)** We examine four R_a values: 13.5, 27, 40.5, 54 Mbps. **(c)** F = 0 Mbps. **(d)** We input estimates of the $y(R_i)$ values which are obtained via comprehensive offline measurements. **(e)** The offline measured PDR_f. We observe that the performance with MRC is quite close to our benchmark measurements. These results show that in spite of having no information with regards to the jammer distribution or the convergence times of the rate adaptation algorithms, MRC is able to significantly help in the presence of a random jammer.

ARES increases the link throughput by up to 150% in an 802.11a mesh deployment with mobile jammers: Next, we apply ARES in an 802.11a mesh network with mobile jammers and UDP traffic. We consider a frequent jammer (jamming distribution $U[1, 20]$ and sleeping distribution $U[0, 1]$). The jammer moves towards the vicinity of the legitimate nodes, remains there for k seconds, and subsequently moves away. For the mobile jammer we used a laptop, equipped with one of our Intel cards, and carried it around. The power control module is implemented in a centralized manner. ARES increases CCA_L in order to overcome the effects of jamming inter-

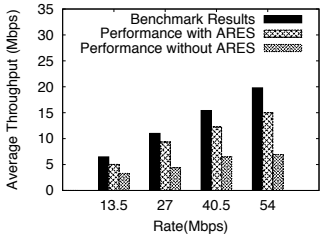


Figure 6.16: ARES provides significant throughput benefits in a MIMO network in the presence of jammers.

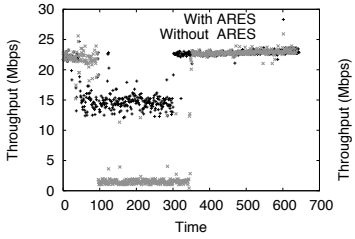


Figure 6.17: ARES provides significant throughput improvement in mobile-jamming scenarios.

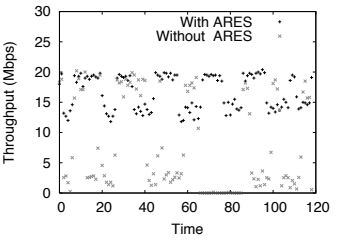


Figure 6.18: ARES improves the client-AP link throughput by 130% with TCP traffic scenarios.

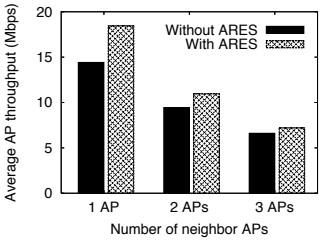


Figure 6.19: MRC improves the throughput of neighbor legitimate devices, as compared to SampleRate.

ference, to the extent possible. In this case, due to the aggressiveness of the considered jammer (prolonged jamming duration), the rate adaptation module does not provide any benefits (since rate control helps only when the jammer is sleeping). In this scenario, ARES follows the steps: $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 6 \rightarrow 7 \rightarrow 8 \rightarrow 9$. Figure 6.17 depicts throughput-time traces, with and without ARES, for an arbitrarily chosen link and $k \approx 200$. The use of ARES tremendously increases the link throughput during the jamming period (by as much as 150 %). We have observed the same behavior with a distributed implementation of the power control module in an 802.11a WLAN setting.

ARES improves the total AP throughput by up to 130% with TCP traffic: Next, we apply ARES on a 802.11a WLAN. For this experiment, we use nodes equipped with the *Intel-2915* cards. We consider a setting with 1 AP and 2 clients, where clients can sense each others' transmissions.

We place a balanced jammer (jamming distribution $U[1,5]$ and sleeping $U[1,8]$) such that all 3 legitimate nodes can sense its presence. We enable fully-saturated uplink TCP traffic from all clients to the AP (using *iperf*) and we measure the total throughput at the AP, once every 0.5 sec. In this scenario, ARES follows the logical steps: $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 6 \rightarrow 7 \rightarrow 8 \rightarrow 9$. From Figure 6.18, we observe that *the total AP throughput is improved by up to 130% during the periods that the jammer is active*. The benefits are less apparent when the jammer is sleeping because TCP's own congestion control algorithm is unable to fully exploit the advantages offered by the fixed rate strategy.

Applying MRC on an AP improves the throughput of neighbor APs by as much as 23%: With MRC, a jammed node utilizes the lowest rate (when the jammer is active) and highest rate (when the jammer is sleeping) that provide the maximum long-term throughput. With this, the jammed node avoids examining the intermediate rates and, as we showed above, this increases the link throughput. We now examine how this rate adaptation strategy affects the performance of neighbor legitimate nodes. We perform experiments on a topology consisting of 4 APs and 8 clients, with 2 clients associated with each AP, all set to 802.11a mode. A balanced jammer with a jamming distribution $U[1,5]$ and a sleep distribution $U[1,6]$ is placed such that affects *only* one of the APs. Only the affected AP is running MRC; the rest of the APs use SampleRate. We activate different numbers of APs at a time, and we enable fully-saturated downlink traffic from the APs to their clients. Figure 6.19 depicts the average total AP throughput. Interestingly, we observe that *the use of MRC on jammed links improves the performance of neighbor APs that are not even affected by the jammer*. This is because the jammed AP does not send any packets using intermediate bit rates (such as with the default operation of rate adaptation algorithms). Since MRC avoids the transmission of packets at lower (that the highest sustained) bit rates, the jammed AP does not occupy the medium for as prolonged periods as with the default rate control techniques; the transmission of packets at the high rate (while the jammer is asleep) takes less time. Hence, this provides more opportunities for neighbor APs to access the medium, thereby increasing the AP throughput. Specifically, we observe that the throughput of one neighbor AP is improved by 23%

(when the topology consists of only 2 APs, one of which is jammed). As we further increase the number of neighbor APs, the benefits due to MRC are less pronounced, due to increased contention (Figure 6.19).

ARES converges relatively quickly: Finally, we perform experiments to assess how quickly the distributed form of ARES converges to a rate and power control setting. In a nutshell, our implementation has demonstrated that the network-wide convergence time of ARES is relatively small. With MRC, the rate control module can very rapidly make a decision with regards to the rate setting; as soon as the jammer is detected, MRC applies the appropriate stored lowest and highest rates.

With regards to the convergence of the power control module, recall that our implementation involves the dissemination of the computed CCA value through the periodic transmission of beacon frames (one beacon frame per 100 msec is transmitted with our ipw2200 driver) [45]. As one might expect, the jammer's signal may collide with beacon frames, and this makes it more difficult for the power control module to converge. Note also that as reported in [45, 106], beacon transmissions are not always timely, especially in conditions of high load and poor-quality links (such as in jamming scenarios). We measure the network-wide convergence time, i.e., the time elapsed from the moment that we activate the jammer until all legitimate devices have adjusted their CCA threshold as per our power control scheme. First, we perform measurements on a multi-hop mesh topology consisting of 5 APs and 10 clients (2 clients per AP). In order to have an idea about whether the observed convergence time is significant, we also perform experiments without jammers, wherein we manually invoke the power control module through a user-level socket interface on one of the APs. We observe that the convergence time for the specific setting is approximately 1.2 sec. Then, we activate a continuously transmitting *deceptive* jammer in a close proximity to 2 neighbor APs (MRC is disabled; the jammer affects only the 2 APs). Table 6.5 contains various average convergence times for the specific setting and for different P_J values.

We observe that although the convergence time increases due to jamming, it still remains short. Furthermore, we perform extensive experiments with 8 APs, 19 clients and 4 balanced jammers

P_J (dB)	Convergence time (sec)
1	1.8
2	2.4
3	2.8
4	3.5

Table 6.5: Average convergence times (in sec) for different P_J values.

with P_J = 3 dBm, all uniformly deployed. We observe that in its distributed form the power control module converges in approximately 16 sec in our network-wide experiments. Although one may expect different (lower or higher) convergence times with different hardware/software and/or mobile jammers, these results show that in a static topology the power control module converges relatively quickly in practical settings.

6.7 Scope of ARES

From our evaluations, it is evident that ARES can provide performance benefits in the presence of jamming, even with other wireless technologies, and both in static and dynamically changing environments. In this section, we discuss some design choices and the applicability requirements of ARES.

ARES does not require additional complicated hardware or software functionalities: The two modules that constitute ARES are relatively easy to implement in the driver/firmware of commodity wireless cards, and do not require any hardware changes. The only software modification needed in the firmware involves the CCA tuning functionality. Specifically, it should be possible to change the CCA threshold as per the commands sent through a driver-firmware socket interface. To facilitate a distributed WLAN implementation of ARES, the AP driver needs to be modified to read the new Beacon template from the Beacons received from neighbor co-channel APs. Finally, clients need to apply the power and CCA settings determined by their affiliated AP.

On the effectiveness of MRC: Our analysis provided in section 6.4 is an accurate tool that decides between the use of a fixed rate or a rate adaptation strategy. However, applying the analysis

in a real system is quite challenging, for various reasons. In particular, as discussed earlier the analysis requires a set of inputs which may not be readily available. If the analysis were to be applied in real time, ARES would need to observe these values on the fly and invoke the rate module whenever significant, non-temporal changes are observed. It is also difficult to derive the jammer's distribution accurately and quickly. Such requirements make the application of the analysis somewhat infeasible in real-time systems. Furthermore, the analysis can account for the presence of one jammer only. In scenarios with multiple jammers, it cannot decide between fixed or adaptive rate.

In contrast, our more practical scheme MRC does not need any inputs. It can operate efficiently even with multiple jammers. Note that MRC in its current form takes into account the time[5] that has elapsed since the last time that rate control was invoked. The policy is to invoke the rate adaptation strategy after periodic intervals. The optimal rate at which rate adaptation should be invoked depends on the temporal variability of the channel. In particular, to perform this optimally, ARES would need to measure (or estimate) the coherence time τ of the channel (time for which the channel remains unchanged [107]) and invoke the rate control algorithm every τ secs. While this is not possible with current 802.11 hardware, it may be possible in the future [107]. Alternatively ARES could employ a learning strategy as discussed in Section 6.5. Enhancing the rate control module to address these issues is in our future plans.

ARES with reactive and constant jammers: For the most part in this work we considered various types of random jammers. With constant jammers, rate adaptation is not expected to provide benefits, since the continuous jamming interference does not allow the use of high rates. Nevertheless, rate control (even as a *standalone* module) is expected to provide benefits in the presence of reactive jamming. In particular, let us consider a link consisting of legitimate nodes A and B. The reactive jammer J needs to sense the ongoing transmission and quickly transmit its jamming signal. If we denote by t_{flight}, the flight time of the legitimate packet and with t_{sense} the time needed for J to sense this packet, then the probability of succesful packet corruption[6] can be calcu-

[5]In its current form, this time is in terms of the number of jamming cycles; this can be easily modified to use more generic time units.

[6]We assume an optimal reactive jammer, i.e., one which is able to jam at the exact time instance when it senses a

lated as: $P_{jam} = P(t_{sense} < t_{flight})$. Assuming that $t_{sensing}$ is uniformly distributed at the interval $[0, DIFS]$[7] we get:

$$P_{jam} = \int_{0}^{t_{flight}} \frac{1}{DIFS}\, dt = \frac{t_{flight}}{DIFS} = \frac{\#bytes/packet}{rate \cdot DIFS} \tag{6.2}$$

From Eq. 6.2 it is clear that through the use of high bit rates and/or reduced packet sizes the probability of succesful reactive jamming can be decreased. However, there is a tradeoff between successful reception and decreased jamming probability that needs to be examined more carefully. Finally, the power control module of ARES, can be useful in the presence of both constant (as shown in the previous section) and reactive jamming.

6.8 Conclusions

We design, implement and evaluate ARES, an anti-jamming system for 802.11 networks. ARES has been built based on observations from extensive measurements on an indoor testbed in the presence of random jammers, and is primarily composed of two modules. The *power control module* tunes the CCA thresholds in order to allow the transmission and capture of legitimate packets in the presence of the jammer's signals, to the extent possible. The *rate control module* decides between fixed or adaptive-rate assignment. We demonstrate the effectiveness of ARES in three different deployments **(a)** a 802.11n based MIMO WLAN, **(b)** an 802.11a network infested with mobile jammers, and **(c)** a 802.11a WLAN with uplink TCP traffic. ARES can be used in conjunction with other jamming mitigation techniques (such as frequency hopping or directional antennas). Overall, the application of ARES leads to significant performance benefits in jamming environments.

Acknowledgments: We thank Ralink Corp. for providing the source of the *RT2860* AP driver, and Dr. Konstantina Papagiannaki from Intel Research for providing the prototype version of the

legitimate packet (best case scenario for the adversary). In reality, this will not be the case.

[7]This is a reasonable assumption to make since the protocol allows for a DIFS period in order to sense any transmission.

ipw2200 driver.

Bibliography

[1] V. Navda, A. Bohra, S. Ganguly, and D. Rubenstein. Using Channel Hopping to Increase 802.11 Resilience to Jamming Attacks. In *IEEE INFOCOM mini-conference*, 2007.

[2] W. Hu, T. Wood, W. Trappe, and Y. Zhang. Channel Surfing and Spatial Retreats: Defenses Against Wireless Denial of Service. In *ACM Workshop on Wireless Security*, 2004.

[3] ISM Wide-band Jammers. http://69.6.206.229/e-commerce-solutions-catalog1.0.4.html.

[4] ISA: Users fear wireless networks for control. http://lists.jammed.com/ISN/2007/05/0122.html.

[5] K. Pelechrinis, C. Koufogiannakis, and S.V. Krishnamurthy. Gaming the Jammer: Is Frequency Hopping Effective? In *WiOpt*, June 2009.

[6] W. Xu, W. Trappe, Y. Zhang, and T. Wood. The Feasibility of Launching and Detecting Jamming Attacks in Wireless Networks. In *ACM MOBIHOC*, 2005.

[7] SESP jammers. http://www.sesp.com/.

[8] W. Hu, K. Ma, W. Trappe, and Y. Zhang. Jamming Sensor Networks: Attacks and Defense Strategies. In *IEEE Network*, May/June 2006.

[9] K. Pelechrinis, G. Yan, S. Eidenbenz, and S. V. Krishnamurthy. Detecting Selfish Exploitation of Carrier Sensing in 802.11 Networks. In *IEEE INFOCOM*, 2009.

[10] M. Heusse, F. Rousseau, G. Berger-Sabbatel, and A. Duda. Performance Anomaly of 802.11b. In *IEEE INFOCOM*, 2003.

[11] I. Broustis, K. Pelechrinis, D. Syrivelis, S.V. Krishnamurthy, and L. Tassiulas. FIJI: Fighting Implicit Jamming in 802.11 WLANs. In *SecureCom*, 2009.

[12] K. Pelechrinis, I. Koutsopoulos, I. Broustis, and S.V. Krishnamurthy. Lighweight Jammer Localization in Wireless Netwroks: System Design and Implementation. In *IEEE GLOBECOM*, 2009.

[13] K. Pelechrinis, I. Broustis, S. V. Krishnamurthy, and C. Gkantsidis. ARES: an Anti-jamming REinforcement System for 802.11 Networks. In *ACM CoNEXT 2009*, 2009.

[14] Jamming attack at hacker conference. http://findarticles.com/p/articles/mi_m0EIN/is_2005_August_2/ai_n14841565.

[15] Techworld news. http://www.techworld.com/mobility/ news/index.cfm?newsid=10941.

[16] RF Jamming Attack. http://manageengine.adventnet.com/ products/wifi-manager/rfjamming-attack.html.

[17] R. Gummadi, D. Wetheral, B. Greenstein, and S. Seshan. Understanding and Mitigating the Impact of RF Interference on 802.11 Networks. In *ACM SIGCOMM*, 2007.

[18] B. O'hara and A. Petrick. *IEEE 802.11 Handbook, a Designer's Companion*. IEEE Press, Second Edition, ISBN 0-73-814449-5.

[19] R. Vedantham, S. Kakumanu, S. Lakshmanan, and R. Sivakumar. Component Based Channel Assignment in Single Radio, Multi-channel Ad Hoc Networks. In *ACM MOBICOM*, 2006.

[20] J. Yee and H. P-Esfahani. Understanding Wireless LAN Performance Tradeoffs. In *http://www.commsdesign.com*, 2002.

[21] P.Li, N.Scalabrino, Y.Fang, E.Gregory, and I.Chlamtac. Channel Interference in IEEE 802.11b. In *Global Telecommunications Conference (GLOBECOM) IEEE*, 2007.

[22] R. Mallik, R. Scholtz, and G. Papavassilopoulos. Analysis of an On-Off Jamming Situation as a Dynamic Game. In *IEEE Trans. Commun., vol. 48, no. 8, pp. 1360-1373*, August 2000.

[23] M.Li, I.Koutsopoulos, and R.Poovendran. Optimal Jamming Attacks and Network Defense Policies in Wireless Sensor Networks. In *IEEE INFOCOM*, 2007.

[24] P. Tague, D Slater, G. Noubir, and R. Poovendran. Linear Programming Models for Jamming Attacks on Network Traffic Flows. In *Network Security Lab (NSL) Technical Report # 002*, 2007.

[25] X.Liu, G.Noubir, R.Sundaram, and S.Tan. SPREAD: Foiling Smart Jammers using Multi-layer Agility. In *IEEE INFOCOM mini-conference*, 2007.

[26] H. Li and Z. Han. Dogfight in Spectrum: Jamming and Anti-Jamming in Multichannel Cognitive Radio Systems. In *IEEE GLOBECOM*, 2009.

[27] R. W. Thomas, R. S. Komali, B. J. Borghetti, and P. Mahonen. A Bayesian Game Analysis of Emulation Attacks in Dynamic Spectrum Access Networks. In *IEEE DySPAN*, 2010.

[28] A.Mishra, V.Shrivastava, S.Banerjee, and W.Arbaugh. Partially overlapped channels not considered harmful. In *SIGMETRICS '06/Performance '06: Proceedings of the joint international conference on Measurement and modeling of computer systems*, 2006.

[29] C.M Cheng, P.H Hsiao, H.T Kung, and D Vlah. Adjacent Channel Interference in Dual-radio 802.11a Nodes and Its Impact on Multi-hop Networking. In *Global Telecommunications Conference (GLOBECOM) IEEE*, 2006.

[30] J.Robinson, K.Papagiannaki, C.Diot, X.Guo, and L.Krishnamurthy. Experimenting with a Multi-Radio Mesh Networking Testbed. In *1st workshop on Wireless Network Measurements (WiNMee 2005), Trento, Italy*, 2005.

[31] V.Angelakis, A.Traganitis, and V.Siris. Adjacent channel interference in a multi-radio wireless mesh node with 802.11a/g interfaces. In *IEEE INFOCOM, poster session*, 2007.

[32] Von Neumann J and O.Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press (May 1, 1980) ISBN 0-69-100362-9.

[33] Soekris-net4826. http://www.soekris.com/net4826.htm.

[34] The MadWiFi driver. http://madwifi-project.org/.

[35] K. Pelechrinis, I. Broustis, T. Salonidis, S. V. Krisnamurthy, and P. Mohapatra. Design and Deployment Considerations for High Performance MIMO Testbeds. In *WICON*, November 2008.

[36] J. Yee and H. P-Esfahani. Understanding Wireless LAN Performance Tradeoffs. In *http://www.commsdesign.com*, 2002.

[37] SNTP, Version 4. http://www.apps.ietf.org/rfc/rfc2030.html.

[38] ioctl() man page. http://linux.die.net/man/2/ioctl.

[39] V.T. Raisinghani and S. Iyer. Architecting Protocol Stack Optimizations on Mobile Devices. In *Cosmoware*, 2006.

[40] V. Navda, A.P. Subramanian, K. Dhanasekaran, A. Timm-Giel, and S. Das. MobiSteer: using steerable beam directional antenna for vehicular network access. In *MobiSys*, 2007.

[41] J. Bicket. Bit-rate Selection in Wireless Networks. In *MS Thesis, Dept. of Electr. Engin. and Comp. Science, MIT*, 2005.

[42] H.Jafarkhani. *Space-Time Coding: Theory and Practice*. Cambridge University Press, 2005.

[43] RT2860 wireless driver. http://www.ralinktech.com/ralink/Home /Support/Linux.html.

[44] V. Mhatre, K. Papagiannaki, and F. Baccelli. Interference Mitigation through Power Control in High Density 802.11 WLANs. In *IEEE INFOCOM*, 2007.

[45] I. Broustis, K. Papagiannaki, S. V. Krishnamurthy, M. Faloutsos, and V. Mhatre. MDG: Measurement-Driven Guidelines for 802.11 WLAN Design. In *ACM MOBICOM*, 2007.

[46] P. Kyasanur and N. Vaidya. Detection and Handling of MAC layer misbehavior in wireless networks. In *DSN*, 2003.

[47] S. Radosavac, J. S. Baras, and I. Koutsopoulos. A framework for MAC protocol misbehavior detection in wireless networks. In *WiSe*, 2005.

[48] M.Raya, J-P.Hubaux, and I.Aad. DOMINO: A System to Detect Greedy Behavior in IEEE 802.11 Hotspot. In *MobiSys*, 2004.

[49] M. Cagalj, S. Ganeriwal, I. Aad, and J.-P. Hubaux. On selfish behavior in CSMA/CA networks. In *INFOCOM*, 2005.

[50] O.Queseth. The effect of selfish behavior in mobile networks using CSMA/CA. In *VTC*, 2005.

[51] J. Konorski. Multiple access in ad hoc wireless LANs with noncooperative stations. In *NETWORKING*, 2002.

[52] IEEE Std 802.11g Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications. 2003.

[53] Ath5k project. http://madwifi.org/wiki/About/ath5k.

[54] GNU radio trac. http://gnuradio.org/trac.

[55] USRP SDR platform. http://www.ettus.com.

[56] J.Lee, S.Choi, and H.Jung. Analysis of User Behavior and Traffic Pattern in a Large-Scale 802.11a/b Network . In *WiNMee*, 2005.

[57] K.Jamienson, B.Hull, A.Miu, and H.Balakrishnan. Understanding the Real-World Performance of Carrier Sense. In *ACM SIGCOMM Workshops*, 2005.

[58] PCAP Unix man page. http://www.tcpdump.org/pcap3_man.html.

[59] Ping Linux Man Page. http://linux.die.net/man/8/ping.

[60] Click Modular Router. http://read.cs.ucla.edu/click/.

[61] S. Zvanovec, P. Pechac, and M. Klepal. Wireless LAN Networks Design: Site Survey or Propagation Models? In *Radioengineering, Vol. 12, No. 4*, Dec. 2003.

[62] T. S. Rappaport. *Wireless communications principles and practices,*. Prentice Hall, 2002.

[63] Path-loss. http://en.wikipedia.org/wiki/Path_loss.

[64] N.R. Draper and H. Smith. *Applied Regression Analysis*. Wiley-Interscience. ISBN 0-471-17082-8.

[65] K. Sundaresan and K. Papagiannaki. The Need for Cross-Layer Information in Access Point Selection Algorithms. In *ACM IMC*, 2006.

[66] B. Kauffmann, F. Baccelli, A. Chainteau, V. Mhatre, K. Papagiannaki, and C. Diot. Measurement-Based Self Organization of Interfering 802.11 Wireless Access Networks. In *IEEE INFOCOM*, 2007.

[67] T. Razafindralambo, I. G. Lassous, L. Iannone, and S. Fdida. Dynamic Packet Aggregation to Solve Performance Anomaly in 802.11 Wireless Networks. In *ACM MSWiM*, October 2006.

[68] H. Kim, S. Yun, I. Kang, and S. Bahk. Resolving 802.11 Performance Anomalies through QoS Differentiation. In *IEEE Comm. Letters, Vol. 9, No. 7*, July 2005.

[69] P. Bellavista, A. Corradi, and L. Foschini. The MUM Middleware to Counteract IEEE 802.11 Performance Anomaly in Context-aware Multimedia Provisioning. In *International Journal of Multimedia and Ubiquitous Engineering, Vol. 2, No. 2*, July 2007.

[70] Hierarchical Token Bucket. http://luxik.cdi.cz/∼devik/qos/htb/.

[71] A. Vlavianos, E. Law, I. Broustis, S. V. Krishnamurthy, and M. Faloutsos. Assessing Link Quality in IEEE 802.11 Wireless Networks: Which is the Right Metric? In *IEEE PIMRC*, 2008.

[72] J. Dunn, M. Neufeld, A. Sheth, D. Grunwald, and J. Bennett. A Practical Cross-Layer Mechanism For Fairness in 802.11 Networks. In *IEEE BROADNETS*, 2004.

[73] M. Portoles, Z. Zhong, and S. Choi. IEEE 802.11 Downlink Traffic Shaping Scheme for Multi-User Service. In *IEEE PIMRC*, 2003.

[74] L. Iannone and S. Fdida. Sdt. 11b: Un Schema a Division de Temps Pour Eviter l'anomalie de la Couche MAC 802.11b. In *CFIP*, April 2005.

[75] S. Yoo, J. Choi, J.-H. Hwang, and C. Yoo. Eliminating the Performance Anomaly of 802.11b. In *ICN*, 2005.

[76] D. Yang et al. Performance Enhancement of Multi-Rate IEEE 802.11 WLANs with Geographically-Scattered Stations. In *IEEE Trans. Mob. Comp., Vol. 5, Iss. 7*, July 2006.

[77] G. Lin and G. Noubir. On Link Layer Denial of Service in Data Wireless LANs. In *Wireless Communications and Mobile Computing*, May 2003.

[78] G. Noubir and G. Lin. Low-power DoS Attacks in Data Wireless LANs and Countermeasures. In *ACM MOBIHOC (poster)*, 2003.

[79] G. Noubir. On Connectivity in Ad Hoc Network under Jamming Using Directional Antennas and Mobility. In *Wired/Wireless Internet Communications, Vol. 2957/2004, pp. 186-200*, 2004.

[80] Rude/Crude measurement tool. http://rude.sourceforge.net/.

[81] A. Akella, G. Judd, S. Seshan, and P. Steenkiste. Self-Management in Chaotic Wireless Deployments. In *ACM MOBICOM*, 2005.

[82] Dueling with Microwave Ovens. http://www.wi-fiplanet.com/tutorials/ article.php/3116531.

[83] Konstantin Gromov, Dennis Akos, Sam Pullen, Per Enge, and Bradford Parkinson. GIDL: Generalized Interference Detection and Localization System. In *ION GPS, Salt Lake City, UT*, 2000.

[84] Liu and Xiangqian. Signal Detection and Jammer Localization in Multipath Channels for Frequency Hopping Communications. In *DTIC*, July 2005.

[85] S.D. Coutts. 3-D jammer localization using out-of-plane multipath. In *RADARCON, Dallas, Texas, USA*, 1998.

[86] E.F. Velez and G.M. Amin. Improved jammer localization using multiple focussing. In *Advanced signal-processing algorithms, architectures, and implementations*, 1990.

[87] A.M. Dean. Detection of active emitters using triangulation and trilateration techniques: Theory and practice. In *AGARD, Radiolocation Techniques*.

[88] I. Broustis, M. Faloutsos, and S.V. Krisnamurthy. Overcoming the Challenges of Security in a Mobile Environment. In *IPCCC, Phoenix, AZ*, 2006.

[89] Y. Cheng, Y. Chawathe, A. LaMarca, and J. Krumm. Accuracy characterization for metropolitan-scale Wi-Fi localization. In *ACM MobiSys, Seattle, WA*, 2005.

[90] A. Savvides, C.C. Han, and M.B. Strivastava. Dynamic fine-grained localization in ad-hoc networks of sensors. In *ACM MobiCom, Rome, IT*, 2001.

[91] A. Subramanian, P. Deshpande, J. Gaojgao, and S. Das. Drive-by localization of roadside WiFi networks. In *IEEE INFOCOM*, 2008.

[92] D. Han, D.G. Andersen, M. Kaminsky, K. Papagiannaki, and S. Seshan. Access Point Localization using Local Signal Strength Gradient. In *PAM*, 2009.

[93] Y. Chen, S. Chen, and W. Trappe. Exploiting Environmental Properties for Wireless Localization and Location Aware Applications. In *PerCom*, 2008.

[94] J. Faruque and A. Helmy. Gradient-Based Routing in Sensor Networks. In *ACM MobiCom (poster session)*, 2003.

[95] R. Poor. Gradient Routing in Ad Hoc Networks. In *www.media.mit.edu/pia/Research/ESP/texts/poorieeepaper.pdf*.

[96] A.P. Ruhil, D.K. Lobiyal, and I. Stojmenovic. Positioned Based Gradient Routing in Mobile Ad Hoc Networks. In *ICDCIT*, 2005.

[97] The UCR testbed. http://networks.cs.ucr.edu/testbed/.

[98] MIT Roofnet. http://pdos.csail.mit.edu/roofnet.

[99] R. Draves, J. Padhye, and B.Zill. Routing in Multi-Radio, Multi-Hop Wireless Mesh Networks. In *ACM MOBICOM*, 2004.

[100] Onoe Rate Control. http://madwifi.org/browser/trunk/ath_rate/onoe.

[101] S. Pal, S. R. Kundu, K. Basu, and S. K. Das. IEEE 802.11 Rate Control Algorithms: Experimentation and Performance Evaluation in Infrastructure Mode. In *PAM*, 2006.

[102] UCR Wireless testbed. http://networks.cs.ucr.edu/testbed.

[103] D. S. J. De Couto, D. Aguayo, J. Bicket, and R. Morris. A High Throughput Path Metric for MultiHop Wireless Routing. In *ACM MOBICOM*, 2003.

[104] J. C. Chen and J. M. Gilbert. Measured Performance of 5-GHz 802.11a Wireless LAN Systems. In *Atheros Comm. White Paper*, August 2001.

[105] SampleRate Bug. http://madwifi.org/ticket/989.

[106] S. Vasudevan et al. Facilitating Access Point Selection in IEEE 802.11 Wireless Networks. In *ACM IMC*, 2005.

[107] J. Camp and E. W. Knightly. Modulation Rate Adaptation in Urban and Vehicular Environments: Cross-Layer Implementation and Experimental Evaluation. In *ACM MOBICOM*, 2008.