ARES: An Anti-jamming REinforcement System for 802.11 Networks

Konstantinos Pelechrinis
Dept. of CS&E
UC Riverside *
kpele@cs.ucr.edu

Ioannis Broustis
Dept. of CS&E
UC Riverside
broustis@cs.ucr.edu

Srikanth V. Krishnamurthy
CS&E, UC Riverside
krish@cs.ucr.edu

Christos Gkantsidis
Microsoft Research
Cambridge, UK
chrisgk@microsoft.com

ABSTRACT

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 paper, 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%.

Categories and Subject Descriptors

C.2.0 [**General**]: Security and Protection; C.2.3 [**Computer Communication Networks**]: Network Operations

General Terms

Design,Experimentation,Measurement,Performance,Security

*This work was done partially with support from the US Army Research Office under the Multi-University Research Initiative (MURI) grants W911NF-07-1-0318 and the NSF NeTS:WN / Cyber trust grant 0721941.

CoNEXT'09, December 1–4, 2009, Rome, Italy.

Keywords

IEEE 802.11, Rate Control, Power Control, Jamming

1. INTRODUCTION

The widespread proliferation of 802.11 wireless networks makes them an attractive target for saboteurs with jamming devices [1, 2, 3, 4]; 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 [5] [6]. Such schemes however, are not effective in scenarios with wideband jammers [7, 8]. 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 [9].

In this paper, 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 [3]. In such cases, ARES can effectively *fight* against the malicious entity, as we discuss later. Our contributions are the following:

[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 3.

1. Understanding the impact of jammers in an 802.11 network with rate/power control. First, 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 [10] 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 [11], Onoe [12] and AMRR (Adaptive Multi Rate Retry) [13]. 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 [10]. This module is used to facilitate successful communications 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 [14]. 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.

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 [14].

Non-stop jamming: *Constant* jammers continuously emit electromagnetic energy on a channel. Nowadays, constant jammers are commercially available and easy to obtain [1, 7]. 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 [14]. 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 [7, 8], 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 [9]. Thus, ARES does not rely on frequency hopping. For a set of related studies based on frequency hopping, please see [5], [6], [15].

Xu *et al.* [14] 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 [16], 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 [17] present an analytical evaluation of the use of cryptographic interleavers with different coding mechanisms to improve the robustness of wireless LANs. In [18], 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 [19] 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.

3. EXPERIMENTAL SETUP

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

Testbed Description: Our testbed consists of 37 Soekris net4826 nodes [20], 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 [21] for the *EMP-8602 6G* cards. We have modified the Linux client driver [22] 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 paper, 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 [23]).

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.

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 [11], AMRR [13] and Onoe [12]. 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. [24]. Table 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 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.

4.1.1 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 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 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 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 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:

$$T_{fixed} = \frac{E[t_s]}{E[t_s]+E[t_j]} \cdot PDR_f \cdot R_s + \frac{E[t_j]}{E[t_s]+E[t_j]} \cdot F, \quad (1)$$

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

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

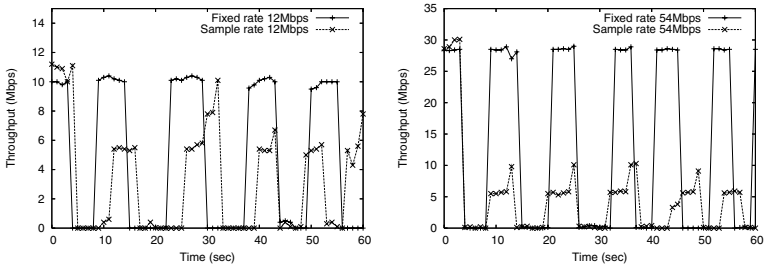

Figure 1: Rate adaptation algorithms may not find the best rate during the sleep period of the jammer. We show cases for $R_a = 12$ Mbps (left) and $R_a = 54$ Mbps (right).

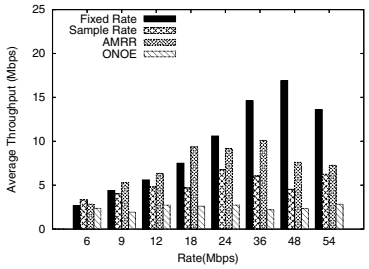

Figure 2: Fixed rates outperform rate adaptation for high-quality links, under random jamming. ($R_a = R_f$)

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 3 plots the values of function y for different values of R_f.

In Table 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 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.

Next, we consider two extreme cases of jamming: frequent and rare jammers (see section 3). The distributions that we use in our experiments for these jammers are shown in Table 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 5 and 6.

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 2: PDR_f thresholds

-	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 3: The jamming distributions that we use in our experiments.

4.1.2 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 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.

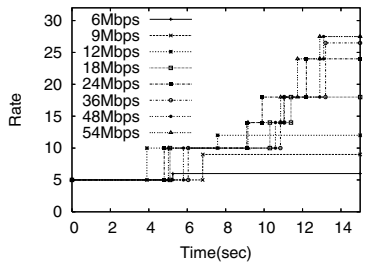

Figure 3: Measured convergence times of the MadWifi SampleRate algorithm, for the different application data rates.

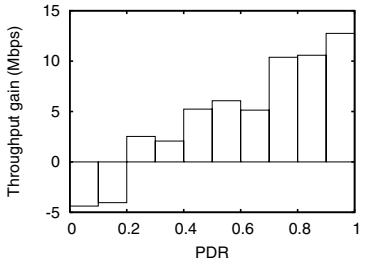

Figure 4: Throughput gain of fixed rate Vs. SampleRate, for various link qualities and for application data rate of 54 Mbps.

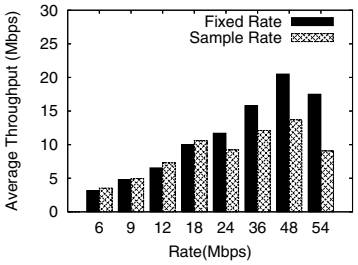

Figure 5: The performance with rare jammers is aligned with our observations for the case with balanced jammers. ($R_a = R_f$)

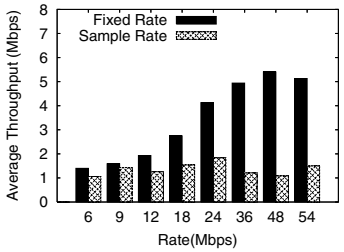

Figure 6: Fixed rate improves the performance more than rate adaptation at high rates, with frequent jammers. ($R_a = R_f$)

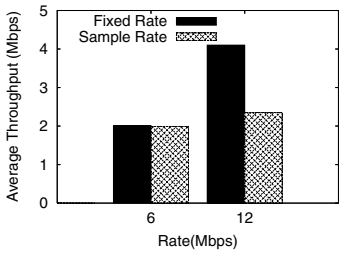

Figure 7: Rate adaptation presents the same behavior in multihop links; it provides lower throughput at high rates.

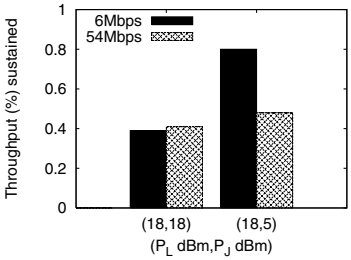

Figure 8: Percentage of the isolated throughput, for various P_L and P_J combinations, for two different transmission rates.

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 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).

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 [17].

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.

4.2.1 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 indicate that when high transmission rates are used, increasing P_L does not help alleviate the impact of jammers. Sample results are depicted in Figure 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 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 4) [10]. 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 4: SINR levels required for successful packet decoding, in 802.11a/g.

4.2.2 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 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 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 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

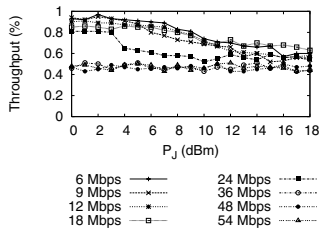

Figure 9: Percentage of the isolated throughput in the presence of a balanced jammer for various P_J and P_J values and data rates.

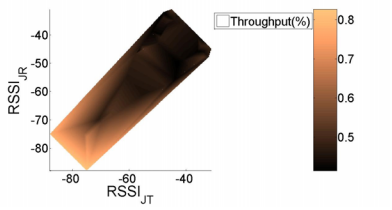

Figure 10: Percentage of the isolated throughput in the presence of a balanced jammer Vs. $RSSI_J$, for CCA_L= –80 dBm.

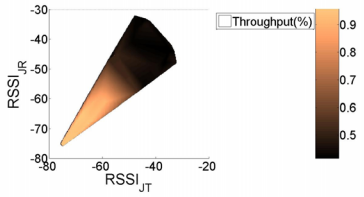

Figure 11: Percentage of the isolated throughput, for various $RSSI_J$ values, and for CCA_L = –50 dBm.

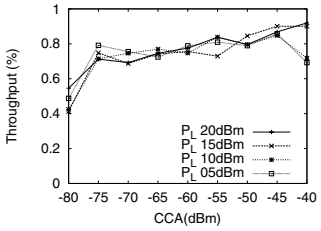

Figure 12: Percentage of the isolated throughput, for various CCA_L values and various P_L values. P_J = 20 dBm.

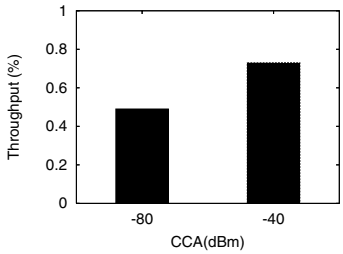

Figure 13: Careful CCA adaptation significantly improves the end-to-end throughput along a route.

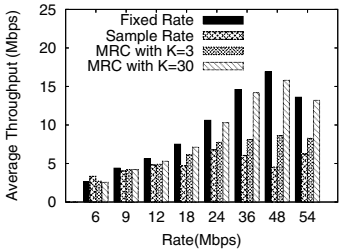

Figure 14: MRC outperforms current rate adaptation algorithms, especially for high values of K.

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 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 12 for $P_L = 5dBm$ and $P_L = 10dBm$.

4.2.3 *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 [10]. 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 [10], [25]. 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 4.1.2. Figure 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.

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 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; [26] 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 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 rescanning 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 paper 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 immediately upon the detection of the next transition from the jammer's active period to the sleeping period.

Figure 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).

Power Control Module in ARES: As discussed in section 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 paper 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 paper.

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 [27]). 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 [25]. 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 chosen 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 [25], [10].

Combining the modules to form ARES: We combine our rate and power control modules to construct ARES as shown in Figure 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 [14] can be used; this functionality performs a consistency check between the

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

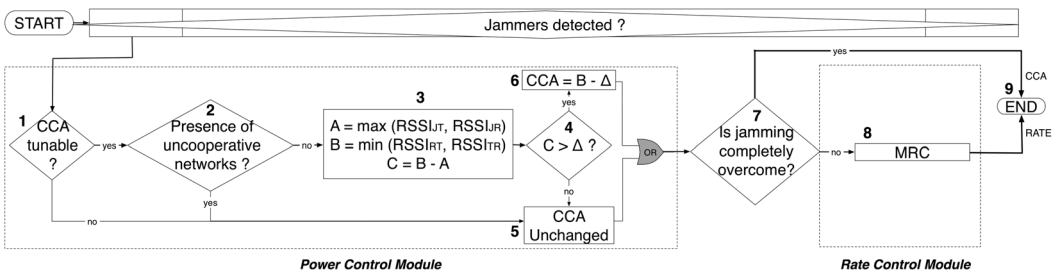

Figure 15: ARES: our Anti-jamming Reinforcement System.

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. 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 [28] 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 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 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 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