On the Robustness of IEEE802.11 Rate Adaptation Algorithms against Smart Jamming *

Guevara Noubir
College of Computer and Information Science
Northeastern University
Boston, MA 02115
noubir@ccs.neu.edu

Rajmohan Rajaraman
College of Computer and Information Science
Northeastern University
Boston, MA 02115
rraj@ccs.neu.edu

Bo Sheng
Dept. of Computer Science
University of Massachusetts
Boston, MA 02125
shengbo@cs.umb.edu

Bishal Thapa
College of Computer and Information Science
Northeastern University
Boston, MA 02115
bthapa@ccs.neu.edu

ABSTRACT

We investigate the resiliency of IEEE802.11 rate adaptation algorithms (RAA) against smart jamming attacks. We consider several classes of state-of-the-art RAAs that include the SampleRate, ONOE, AMRR, and the RAA used in Atheros Microsoft Windows XP driver. We model the behavior of these algorithms, and show the existence of very efficient attacks that exploit RAA-specific vulnerabilities as well as the inherent weaknesses that exist in the design of IEEE802.11 MAC and link layer protocol: in particular the overt packet rate information being transmitted, predictable rate selection mechanism, performance anomaly caused by the equiprobability of transmissions among all nodes regardless of the data rates being employed, and the lack of interference differentiation from poor link quality by IEEE802.11 RAAs. In this work, we present algorithms that determine optimal jamming strategies against RAAs for a given jamming budget, and experimentally demonstrate the efficiency of these smart jamming attacks, which can be orders of magnitude more efficient than naive jamming. For example, in the case of SampleRate, eight reactive jamming pulses every second are sufficient to achieve the same network throughput degradation achieved by a periodic jammer with the jamming energy cost 100 times higher. Some of the RAAs react even worse to smart jamming attacks; ONOE in particular suffers from the phenomenon of *congestion collapse* where the nodes fail to recover from the lowest data rate even after the jammer stops jamming. At the end, we summarize fundamental reasons behind such RAA vulnerabilities and propose a preliminary set of mitigation techniques. We leave the experimental demonstration of the efficiency of the proposed mitigation mechanisms for future work.

*This work was partially supported by NSF grant 0915985.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
WiSec'11, June14–17, 2011, Hamburg, Germany.
Copyright 2011 ACM 978-1-4503-0692-8/11/06 ...$10.00.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design —*Wireless Communication*

General Terms

Security, Algorithms

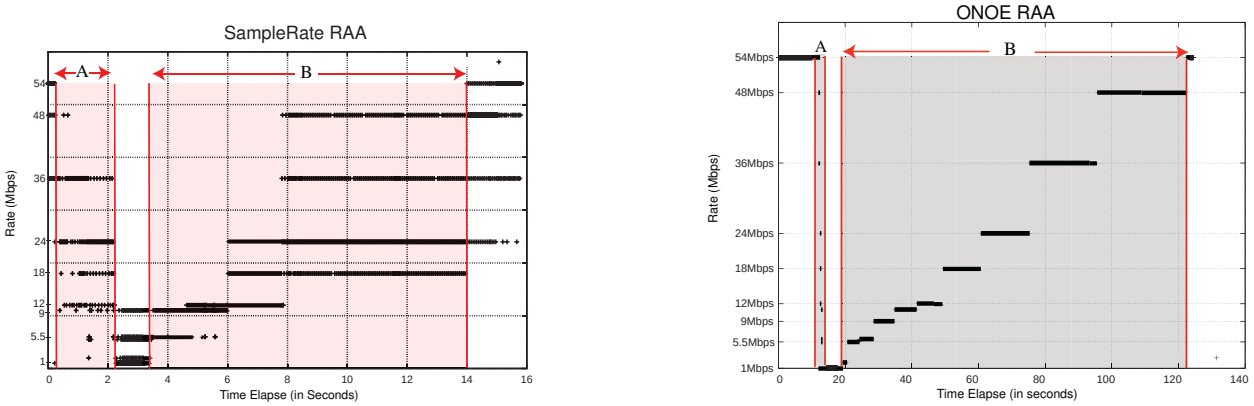
Keywords

IEEE802.11, Rate Adaptation, Smart Jamming, USRP, GNU Radio, Experimentation

1. INTRODUCTION

With rapid advancement and standardization of wireless technology, wireless LANs (WLANs) are now ubiquitous, providing the last mile access to the Internet. Security issues in WLANs, however, remain a serious concern and have attracted a lot of attention in the research community. Among various security attacks, jamming continues to be an effective exploit that can deny or degrade service to legitimate WLAN users. A knowledgeable attacker can intermittently inject signals into the medium and occupy wireless channels, interfere with regular traffic and disrupt WLAN operations effectively at minimal jamming cost. Existing jammers rely on high transmission power and frequent injection of jamming signals to disrupt communication. Such a strategy is inefficient in terms of jamming power consumption; furthermore, increases the risk of trivial jammer detection due to high power jamming and/or frequent packet injections into the communication medium.

In this paper, we study the resiliency of IEEE802.11 rate adaptation algorithms (RAAs) against smart jamming attacks. To this end, we first consider the design of an optimal jammer targeting the vulnerabilities of IEEE802.11 RAAs. The function of the RAA is to enable WLAN users to adaptively choose the best transmitting rate according to current wireless link conditions in order to achieve the maximum throughput possible. Intuitively, lower rates are more reliable and suitable for poor channel conditions and the higher data rates for good channel conditions. It is well known that most common implementations of RAA in use today cannot distinguish between the causes of packet failures due to the poor link quality and due to the interference/collisions. If a jammer injects



(a) Samplerate (b) ONOE

Figure 1: (a) Time it takes to drop the rate to the lowest value, (b) Time it takes to recover to the optimal data rate.

pulses so as to interfere with the regular packets, the legitimate user will assume that the link quality is poor for the current transmission rate, and will decrease the rate to a lesser value. What makes it worse is that once the jamming stops, the recovery period before attaining the optimal data rate for communication in these RAAs are much longer than the time it takes for them to adaptively lower the data rate due to collision or malicious interference. Figure 1 shows such a phenomenon that is predominant in the case of two of the most commonly used RAAs: SampleRate and ONOE. Overall, this is one of the reasons that the IEEE802.11 wireless network suffers from significant performance degradation even with an oblivious jammer injecting small pulses intermittently into the channel.

IEEE-802.11 WLAN also suffers from a performance anomaly where the poorest link dictates the throughput of the whole network sharing a common channel. This phenomenon was first reported in [11]. The reason for this is because all participating nodes have the same probability of transmitting at any instant of IEEE802.11 communication independent of the data rates being used at different links. However, this behavior leads to an efficient adversary attack called the *reflection attack* where the jammer targets one particular link and jams to bring down its data rate, while in effect causing the whole network throughput to suffer heavily as shown in Figure 2. Figure 2(a) depicts an experimental setting with multiple links sharing a common channel and a reactive jammer present in the medium that selectively jams all the non-1Mbps traffic of some link, l_i. Without the loss of generality, we pick D as the victim link for the experiment. Figure 2(b) clearly shows that the impact of jamming victim link D trickles down to non-victim links (A, B, C, D) in terms of the average throughput degradation and ultimately the whole network throughput is affected even though only a single link is being targeted by the jammer.

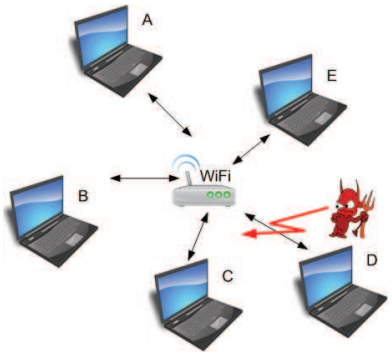
Hence, in this paper, we carefully analyze the vulnerabilities inherent to IEEE802.11 MAC and RAAs, and design optimal jammer exploits to maximize the throughput reduction at a minimal jamming cost. The main contributions of our work are as follow:

- We first analyze three widely-used RAAs – ONOE, AMRR, SampleRate and derive the cost of jamming in each case to achieve a desired throughput reduction (Section 5).
- We then classify RAAs based on their rate selection strategies and use that framework to design optimal jamming strategies that exploit the RAA-specific behavior. We show that our jamming cost analysis can be used to efficiently design a smart jammer that targets specific packets and optimize the reduction in throughput when subjected to a jamming energy budget. We also present a technique that applies to the case when jamming costs cannot be estimated (Section 4).
- We carefully analyze the weaknesses inherent in the IEEE802.11 MAC and Link layer protocols that allow jammers to be extremely efficient with their jamming.
- We build a testbed comprising of the USRP/GNURadio platform and present a comprehensive experimental evaluation of three RAAs mentioned above and the RAA used in Microsoft Windows XP in the presence of smart jamming. Our experimental results confirm that a smart reactive jammer can maintain links at a low data rate (1 Mbps) at a minimal jamming cost (jamming only $5 - 8$ packets/s) (Section 6).
- Finally, we propose a set of preliminary mitigation mechanisms with their implementation left for future work (Section 7).

Paper Outline: In Section 2, we review the related work. In Section 3, we provide the background for our work and discuss inherent weaknesses of IEEE802.11. Subsequently, we discuss our system model and the framework for cost analysis of IEEE802.11 RAA jamming in Sections 4 and 5. Finally, in Section 6, we evaluate the proposed smart jamming attacks using a carefully designed real-world experimentation test-bed. We present the details of the implementation and evaluation methodology. At the end, we conclude with the discussions on the mitigation techniques and future work in Sections 7 and 8.

2. RELATED WORK

Anti-jamming techniques have been studied extensively for decades [30]. Most of the earlier mechanisms focused on protecting physical layer of the wireless communication and made use of spread-spectrum techniques, directional antennas, and coding schemes. At the time, most of the wireless communication were not packetized nor networked. Reliable communication in the presence of adversaries have regained significant interest in the last few years. New attacks and thus the need for more complex applications and deployment environments have emerged. Several specifically crafted attacks and counter-attacks have been proposed for: packetized wireless data networks [20, 22], multiple access resolution in the presence of adversaries [1–3], multi-hop networks [20,



(a) Reflection Attack

	Victim Link	Non-victim Links
Avg. Throughput (Pre-jamming)	16.1 Mbps	15.4 Mbps
Avg. Throughput (Post-jamming)	1.02 Mbps	0.96 Mbps
Data Rate (Pre-jamming)	54 Mbps	54 Mbps
Data Rate (Post-jamming)	1 Mbps	54 Mbps

(b) Performance Anomaly

Figure 2: Smart Jamming Strategy that exploits the IEEE802.11 Performance Anomaly.

33,43], broadcast communication [8,10,32], cross-layer attacks [21], and navigation information broadcast systems [27]. However, very little work has been done on protecting rate adaptation algorithms against adversarial attacks. Rate adaptation plays an important role in IEEE802.11 as the link quality in a WLAN is often highly dynamic. In recent years, a number of algorithms for rate adaptation have been proposed in literature [7, 12, 14, 17, 25, 26, 41, 42], and few have been used in Commercial Off-The-Shelf (COTS) products as well [5, 19]. Their main idea is to estimate channel quality and adjust the transmission rate accordingly. Most of the existing mechanisms estimate channel quality using some metrics, such as statistics of packet successes and failures, PHY metrics like the SNR, probe packets etc. Based on where this information is collected, we can classify RAAs into two main categories: the first category is *sender-based* where the RAAs determine the rates solely based on the metrics collected and calculated at the sender side (irrespective of the receiver feedback/information) [5, 15, 19, 26, 42] and the second category is *receiver-based* where the algorithm explicitly uses feedbacks from the receiver to select a data transmission rate [9,12,17,28,29]. Regardless, most of these algorithms are vulnerable to even the simplest of jamming attacks mainly because these algorithms fail to differentiate between interference caused by link quality and collisions/fading/malicious interference, and therefore causing the collected statistical information to be biased by the interference making them an inaccurate assessment of the link quality. [17] tries to differentiate between the losses due to collision and link conditions using RTS/CTS exchange. The basic intuition in their work is that with the RTS/CTS enabled, the packet loss is certainly caused by the link quality. Robust Rate Adaptation Algorithm in [42] adopts a similar idea to obtain more accurate statistics of packet loss. However, these approaches cannot mitigate the issue of (malicious) interference caused by jamming because the adversary may not obey the RTS/CTS policy, e.g., the jammer can jam the data packet following an RTS/CTS exchange.

[28] proposes to let the receiver send corrupted packets back to the sender to help determine the cause of the packet failure. However, it does not help the WLAN under jamming attacks either because, (a) the interference caused by jamming may have different characteristics from the interference caused by the channel noise. So it is difficult for the proposed scheme to detect the existence of the jamming signals, and (b) the adversary may jam the feedback packets from the receiver so that the sender has no sufficient information for analysis.

To the best of our knowledge, [23] is the first work to consider RAA jamming. They demonstrate that in fact existing RAAs are highly vulnerable to jamming. They show that fixed data rate network outperforms most of the rate-adaptive network in the presence of naive jamming. Their work, however, assumes an unconstrained jammer and does not consider the case where the jammers, similar to the senders and receivers, are resource constrained. In contrast, our paper focuses on the robustness of RAAs against attacks that specifically target the specific rate adaptation vulnerabilities and IEEE802.11 weakness and efficiently reduce the network throughput within their limited jamming budget. Furthermore, their anti-jamming mechanism depends on figuring out correct threshold to distinguish between scenarios of jamming and no-jamming, and hence to be able to switch between using RAA and fixed data rate as triggered by their threshold cutoff. This scheme of calculating the appropriate threshold can easily be exploited by a smart jammer. Their work, therefore, only provides resiliency against naive jamming. Our work, on the other hand, discusses the mitigation of vulnerabilities at the IEEE802.11 MAC and Link Layer to keep smart jammers from launching efficient denial of service attacks.

Lastly, [6] considers intelligent jamming that exploits the performance anomaly in IEEE802.11 WLAN. They propose ways to detect and alleviate the impact of such jamming under their setup. Their work, however, like [23] only considers inefficient jammers that blindly send intermittent or periodic signals without exploiting the publicly known protocol information. Therefore, their jammer requires being physically placed in the vicinity of the victim node such that it does not jam a region and instead jams only the targeted victim node. Jamming a region is detrimental to the jammer's performance not only in terms of jamming cost but also it would lead to easy detection. In contrast, our work considers reactive jammers that are not only channel aware but specifically filter out the victim nodes traffic, and focus all of their jamming on the packets of the victim node as shown in Figure 2.

3. IEEE802.11 MAC AND RAAS

In this section, we briefly introduce the specific RAAs that we analyze in this paper, and discuss the inherent weaknesses of IEEE-802.11 MAC and RAAs that allow for smart jamming attacks.

3.1 Background

We have examined four RAAs in our experiments. The first three are included in MadWifi driver with source codes and the fourth one is the Atheros Windows XP driver for which details are not available. The following is a brief description of the Madwifi (most popular open source driver) implementation of the RAAs [36]:

SampleRate: SampleRate is the default RAA used in Madwifi driver [5, 35, 36]. It maintains the statistical information for each data rate which includes the average transmitting time (ATT) (considering the retries) and the number of consecutive failures. The algorithm picks the rate with the lowest ATT for transmission. After every 10 packets, SampleRate randomly picks a different data rate for probing to update its packet rate statistics. This allows for rate update even when the current rate is performing with no failures. The ratesbe with more than 3 consecutive failures are not eligible for probing (black-listed). After a 2 second period, all the black-listed rates are reconsidered for probing if their theoretical

ATT with no retransmissions is better than the ATT of the current rate in use. In non-malicious environment, SampleRate quickly converges to the optimal data rate.

ONOE: ONOE is the RAA developed by the Madwifi developers [37]. ONOE monitors the history of packet successes and failures within a window of 1 second, and uses a credit system where it increases the credit value by 1 if more than 90% of packets have succeeded during the last window, otherwise decreases the credit by 1. When the credit reaches 10, the sender increases the transmitting rate to the next higher rate. If all the packets succeed during the last monitor window, then the sender directly increases the rate. If each packet fails at least once on average, then the sender decreases the rate to the next lower rate. Therefore, Onoe is more conservative in its step-ups as it takes at least 10 seconds before it decides to increase the data rate, whereas, it steps down pretty quickly if the link quality deteriorates.

AMRR: Adaptive Multi Rate Retry is a two-stage RAA, which is basically an extension of ARF (Auto Rate Fallback) with multi-rate retransmissions [19]. The main idea behind this RAA is that the short-term fluctuations are dealt via multi-rate retries (MRR) implemented at the driver, and the long term rate adaptation is taken care of by applying a basic mechanism where the sender adjusts the rate upwards after 10 consecutive ACKs, and adjusts the rate downwards after 2 consecutive failures. The MRR is defined as a tuple (r_0/c_0, r_1/c_1, r_2/c_2, r_3/c_3), where the retry rates (r_0, r_1, r_2, r_3) are set to $(the\,current\,rate,\ one\,level\,lower\,bit - rate\,,\ two\,level\ lower\,bit - rate,\ the\,lowest\,base\,rate)$ and the retry counts (c_0, c_1, c_2, c_3) are set to $(1, 1, 1, 1)$ respectively. The more details on the multi rate retries for AMRR, ONOE and SampleRate can be found in [18].

3.2 Weaknesses of IEEE802.11 MAC and RAAs

There are four major weaknesses in existing rate adaptation algorithms used in combination with IEEE802.11 Link and MAC layer protocols for WiFi communication:

- *Overt Packet Rate Information:* The IEEE802.11 standard makes the rate of the current packet being transmitted explicitly available in the SIGNAL field of the PLCP header (encoded and modulated with a robust base rate). This allows an adversary equipped with a smart radio to quickly identify the current packet rate and jam it before the end of the transmission. Even without the PLCP header information, a smart jammer can recover the rate of a packet by (1) analyzing the I and Q signal constellation to derive the current modulation of the packet (e.g., BPSK, QPSK, 16QAM, 64QAM for 802.11ag), and (2) attempting the error correction schemes (e.g., 1/2, 2/3, 3/4 for 802.11g). The combination of modulation and coding scheme uniquely identifies the packet rate in a 802.11 communication. Furthermore, distinguishing between DSSS and OFDM are even easier when using the spectrum signature of the frame preamble. This allows for an easy detection of current data rate in use by an adversary.

- *Predictable rate selection rules:* The behavior of existing rate adaptation algorithms is very predictable. SampleRate, for example, sends probes periodically at the interval of every ten packets with a different data rate. Furthermore, the Madwifi (most widely used linux driver) implementation of SampleRate makes it even more predictable by using deterministic rules to pick data rates for probing. Similarly, the credit mechanisms of ONOE is easily track-able by an adversary, and so is the exponential backoff mechanism of AMRR.

- *Equi-probable transmissions:* The IEEE802.11 standard gives equal opportunity to all the nodes to transmit, independent of their link quality (therefore the data rates). This allows adversaries to mount *reflection attack*, where a victim node is targeted for attack and forced into selecting a low data rate. The victim now monopolizes the channel, therefore indirectly blocking/delaying other nodes from transmitting. This can transform moderate load into saturation traffic load, and possibly a self-sustaining low-rate selection phenomenon as seen in the case of ONOE. We call this jammer-triggered *congestion collapse*. We discuss this in detail in Section 5.

- *Lack of Interference Differentiation:* Radio receivers are incapable of differentiating between *malicious interference* (e.g., jamming) and *non-malicious interference* such as direct collisions (two nodes' MAC backoff timer expiring at the same time), hidden terminal problem, or noise from spatial reuse of channels. This gets even harder to do for moving nodes with dynamic link quality due to multi-path fading and environmental changes.

All these weaknesses in IEEE802.11 MAC and RAAs allow for very efficient and effective attacks by an adversary spending minimum jamming cost.

4. IEEE802.11 RAA JAMMING

In this section, we present our model for studying RAA jamming attacks and devise optimal attack against rate adaptation given a fixed energy budget for the jammer.

4.1 System Model and Problem Formulation

4.1.1 Network Model

We consider a WLAN with a set U of users who share the same wireless channel and are all within one another's communication range. We assume there are n directional communication links given by the set $L = \{l_1, l_2, \ldots, l_n\}$ among all users. Let $D = \{d_i > 0 | 1 \leq i \leq n\}$ denote the set of expected traffic demands on links L. For a saturated network, $d_i = d$ for all links l_i. Let m represent the available number of transmitting data rates for each user and R denote the set of data rates in ascending order:

$$R = \{r_1, r_2, \ldots, r_m\} \ \forall i, j, i < j \Rightarrow r_i < r_j.$$

Now, we use $t_i \in R$ to represent the transmission rate used on the link l_i (i.e, by the sender). Thus, $T = \{t_1, \ldots, t_n\}$ represents the array of rates used on the n links respectively. The overall throughput of the WLAN can be approximately expressed as

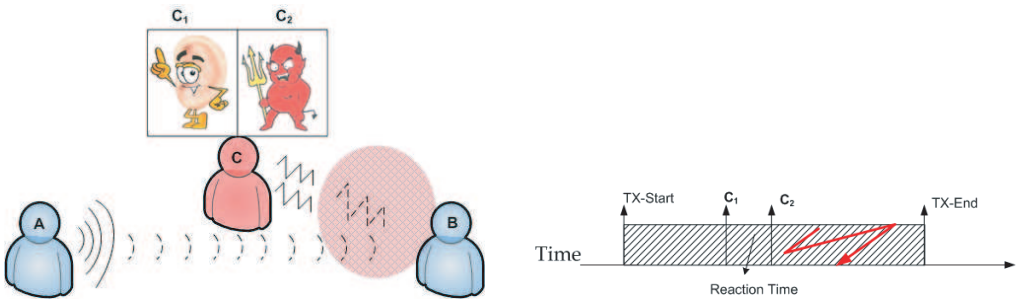
$$\Gamma(D, T) = \frac{\sum_i d}{\sum_i \frac{d}{t_i}}.$$

$$\iff \Gamma(D, T) = \frac{1}{\sum_i \frac{1}{t_i}}$$

Note that L, D and T may vary over time. Here, we assume that they are stable for an epoch t.

4.1.2 Adversary Model

We assume the adversary is equipped with a radio device operating on the 802.11 frequency band that can receive signals from the air and inject signals to it. Under our model, the jammer uses a fixed transmission power to generate a short pulse signal that is strong enough to jam a packet if hit. During our experimental evaluation, we observed that a jamming pulse as small as $22\mu s$ in length



(a) Network Setup (b) Timeline Graph

Figure 3: Block Diagram: A is the sender, B is the receiver, C is the jammer (C_1 and C_2 are its sensing and jamming counterparts).

can make the concurrently transmitted packet undecodable even if it is being sent using the most reliable rate by the sender. Additionally, we assume that the adversary has a bounded energy source which limits it to injecting at most B jamming packets during any epoch. Thus, the objective of the adversary is to jam the WLAN with at most B packets and alter the transmitting rates, T, to a new set T' such that $\Gamma(D, T')$ is minimized.

Figure 3 depicts the network and the adversary model of our system.

4.2 Sketch of an Attack

We divide each jamming attack into two phases, an *initial phase* and a *maintenance phase*. The first phase is to bring down the transmitting rates on some links and the second phase is to keep those links at the low rate for the duration of the epoch given a fixed jamming budget.

4.2.1 Initial Phase

This phase is a short period compared to the maintenance phase and the epoch. We assume that prior to the start of the initial phase the adversary has monitored the traffic in the air and obtained some information about the WLAN, such as the identifier of each user and data link, traffic demands (length of the payload in each link), and current rates. In this initial phase, the adversary first selects a set of victim links and calculates a target rate for each of them that is no more than the current rate. Then, if the target rate for a victim link is smaller than the current rate, the adversary intensively jams the packets transferred on the victim link to trigger the RAA to decrease the data rate. The jamming in this phase stops when the rate on each victim link reaches the target level. Our goal in this phase is to bring down the transmitting rates on the victim links to the target rates in a quick and efficient way. As we will see later, it may not be necessary to jam all the packets on the victim links to achieve this phenomenon.

4.2.2 Maintenance Phase

After the initial phase, each victim link's rate has already been decreased to the adversary's target rate. However, additional jamming is needed to prevent those links from recovering to their previous higher rates as more packets are being delivered. We call this the maintenance phase. In this phase, the adversary selectively jams the packets transferred on the victim links so that the RAA does not increase the rates. Compared to the initial phase, the jamming in the maintenance phase is less frequent, but it lasts for a longer period (the remainder of the epoch) depending on the budget and the goal of the jammer.

In this two-phase attack design, the challenge is to determine which victim links to invest the energy on and achieving their corresponding target rates. In the rest of this section, we present algorithms to solve this problem. We look at two scenarios differentiated based on the awareness or the oblivion of the adversary regarding the RAA being employed by communicating links.

Fig. 4 shows the performance of RAA jamming in terms of above described phase costs for each of the four RAAs (SampleRate, ONOE, AMRR, Windows RAA).

4.3 Optimal Algorithm for Known RAA

We first consider the case where the jammer is aware of the RAA being used for each link in the network.[1] With the RAA information, we can derive two important parameters of the jamming cost spent in the initial phase and maintenance phase. We call them *initial cost* and *maintenance cost*, which are defined as the number of jamming pulses needed in the initial and maintenance phases under our model respectively. Let $ic(r, r', k)$ denote the initial jamming cost of degrading the transmitting rate on link l_k from r to r' ($ic(r, r', k) = 0$ if $r \leq r'$). Let $mc(r, k)$ denote the maintenance cost of keeping link l_k at rate r for the duration of an epoch. In the next section, we will show how to calculate them $ic(r, r', k)$ and $mc(i, k)$ for a particular RAA. Here, we assume that they are given parameters for the problem.

Given the demand D, and the set of initial transmission rates T, our goal is to find the best jamming strategy to yield a new set of rates T' and maintain it for the epoch t within the jammer budget B[2]. Thus, our problem can be formulated as

$$\text{minimize } \Gamma(D, T') \Rightarrow \text{maximize } \sum \frac{1}{r'_i}$$

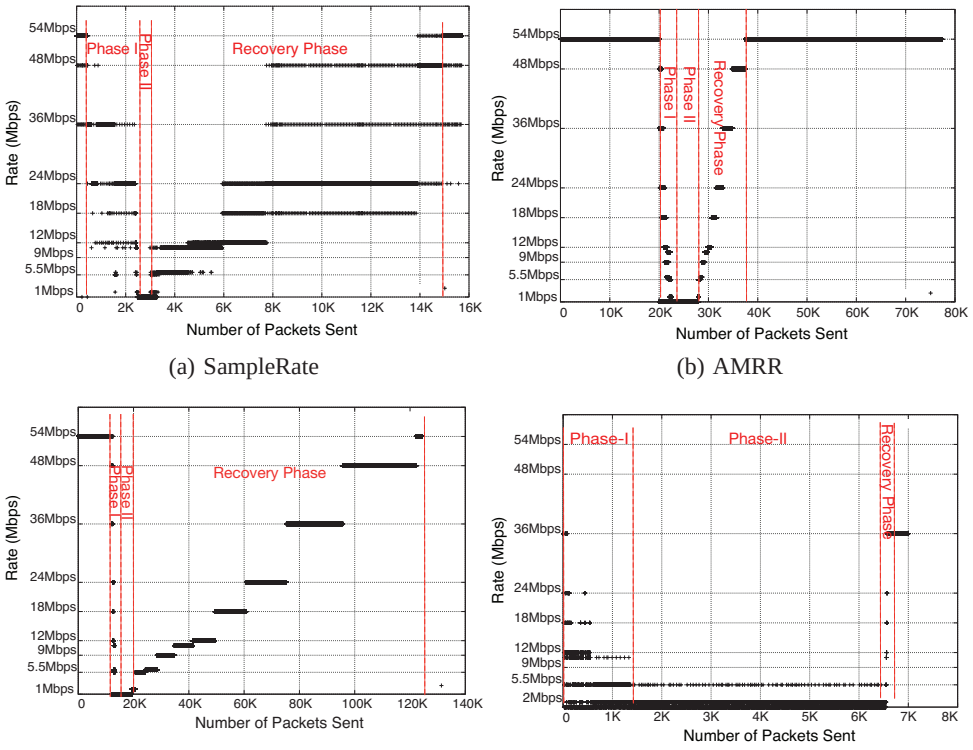
$$s.t. \quad \sum_{\forall k} (ic(t_k, t'_k, k) + mc(t'_k, k)) \leq B$$

We propose Algorithm 1 to find the best victim links and their target rates. In the algorithm, $Opt(x)$ represents the maximum objective we can achieve with cost budget x and $T^x = \{t_1^x, t_2^x, \ldots, t_k^x\}$ is the corresponding resulting set of rates, i.e., link l_k uses rate t_k^x in the optimal result.

Initially, Opt is set to the current value of $\sum \frac{1}{t_i}$. The algorithm is a dynamic program that incrementally fills the array $Opt(x)$. Lines 4-17 enumerate all possible choices for the last jamming action which can be represented as a $< l_k, r, r' >$ tuple, i.e., the last jamming to decrease the rate on l_k from r to r'. In line 7, y represents the budget cost for all the previous jamming actions. $< l_k, r, r' >$ is valid only if $y > 0$ and given that the jamming link l_k was using rate r. Among all possibilities for the last jamming action, we

[1]Note each link may use a different RAA.

[2]Note that while demands might be difficult to predict, it can be assumed to be uniform in saturated conditions.



(a) SampleRate (b) AMRR

(c) ONOE (d) Windows

Figure 4: Performance of RAAs under Jamming

Algorithm 1 Optimal Jamming for Known RAA

1: $Opt(0) \leftarrow \frac{\sum d_i}{\Gamma(D,T)}$, $T^0 \leftarrow T$
2: **for** $x = 1$ to B **do**
3: $\quad Opt(x) \leftarrow Opt(x-1), T^x \leftarrow T^{x-1}$
4: $\quad$ **for all** link l_k **do**
5: $\quad\quad$ **for** r in R such that $r \leq t_k$ **do**
6: $\quad\quad\quad$ **for** r' in R such that $r' < r$ **do**
7: $\quad\quad\quad\quad y \leftarrow x - ic(r, r', k) - mc(r', k)$
8: $\quad\quad\quad\quad$ **if** $y > 0$ **then**
9: $\quad\quad\quad\quad\quad ty \leftarrow Opt(y) - \frac{1}{r} + \frac{1}{r'}$
10: $\quad\quad\quad\quad\quad$ **if** $t_k^y = r$ and $ty > Opt(x)$ **then**
11: $\quad\quad\quad\quad\quad\quad Opt(x) \leftarrow ty$
12: $\quad\quad\quad\quad\quad\quad T^x \leftarrow T^y$ and set $t_k^x \leftarrow r'$
13: $\quad\quad\quad\quad\quad$ **end if**
14: $\quad\quad\quad\quad$ **end if**
15: $\quad\quad\quad$ **end for**
16: $\quad\quad$ **end for**
17: $\quad$ **end for**
18: **end for**
19: Output $Opt(B)$ and $T' \leftarrow T^B$

pick the one that can maximize the objective and record the maximum value in $Opt(x)$ (lines 11-12). T^x is also updated to keep track of the victim links and their target rates. Finally, T^B contains the best target rates for each victim link and $Opt(B)$ is the optimal value of the objective function. The complexity of Algorithm 1 is $O(B \cdot n \cdot m^2)$.

4.4 Greedy Algorithm for Unknown RAA

If the RAA in the WLAN is unknown, the adversary may not be able to calculate the initial cost and maintenance cost. An alternative way is to run a short training session where the adversary conducts jamming trials on each link at different rates and estimates the initial cost and maintenance cost based on the observation. If the training session cannot provide a consistent and accurate estimation, we propose the following greedy heuristic for the reactive jammer.

At all times, for each link l_i the adversary maintains an estimate gmc_i of the cost of maintaining the link at its current rate for the duration of the epoch. It also maintains the total cost gic incurred thus far in bringing the rates of the victim links down to the current rates. Initially, gmc_i for all i and gic are set to zero.

To determine the next victim link, the adversary computes, for each link l_i, the degradation in throughput, $GD(i)$, when its rate is decreased by a level (assuming it is not already at the lowest level).

$$GD(i) = d \cdot \left(\frac{1}{p(t_i)} - \frac{1}{t_i} \right),$$

where for a given rate r, $p(r)$ is the largest rate smaller than r in R; if r equals r_1, then $p(r)$ is also r_1. If $gic + \sum_i gmc_i < B$, the adversary selects the link l_i with the maximum value of $GD(i)$ as a victim link and jams the link until l_i starts to use $p(t_i)$. The cost gic is accordingly updated to include the number of packets used in the preceding step. Then, the adversary updates the estimate gmc_i by simply jamming packets higher than the current rate and extrapolating the cost over the epoch duration.

The adversary repeats this process to identify multiple victim links for the epoch until the budget is exhausted. The greedy algorithm is summarized below.

Algorithm 2 Greedy Algorithm for Unknown RAA

1: $gic = 0, \forall i\ gmc_i = 0, J = \{\}$
2: **while** $B - gic - \sum_i gmc_i > 0$ **do**
3: Pick link l_i where $i = \arg\max GD(i)$; if no such link exists, then exit the loop
4: $J \leftarrow J + \{l_i\}$
5: Jam l_i until B' packets are injected or rate of l_i reduced by one level (assume x packets are injected)
6: Monitor a time unit to update the cost estimate gmc_i for maintaining l_i at current rate for epoch
7: $gic \leftarrow gic + x$
8: **end while**
9: Launch maintenance jamming on all links in J

We can prove that the greedy algorithm is, in fact, optimal if the following two conditions hold true:

1. RAA satisfies a special property: the *forward difference*[3] of rate reciprocal sequence (in increasing order) is non-decreasing, i.e.,

$$\frac{1}{r_i} - \frac{1}{r_{i+1}} \geq \frac{1}{j} - \frac{1}{r_{j+1}}\ where\ 1 \leq i \leq j \leq n$$

2. The maintenance cost is uniform across all links and rate levels.

The first condition implies that the incremental performance degradation increases with a decrease in rate level. This applies to all the IEEE802.11 variants studied in this paper. In the next section, we show that the second condition also approximately holds for all of the RAAs studied in this paper.

5. JAMMING COST FOR KNOWN RAA

In this section, we analyze the initial and maintenance costs for jamming RAAs. As discussed in the preceding section, an adversary equipped with such an analysis can efficiently design an optimal jammer. Here we only consider the three RAAs – ONOE, AMRR, and SampleRate – used in our experiments. For other RAAs, similar analysis can be conducted. AHence, in this paper, we carefully analyze the vulnerabilities inherent to IEEE802.11 MAC and RAAs, and design optimal jammer exploits to maximize the throughput reduction at a minimal jamming cost. The main contributions of our work are as follow:

- We first analyse three widely-used RAAs – ONOE, AMRR, SampleRate and derive the cost of jamming in each case to achieve a desired throughput reduction (Section 5).
- We then classify RAAs based on their rate selection strategies and use that framework to design optimal jamming strategies that exploit the RAA-specific behaviour. We show that our jamming cost analysis can be used to efficiently design a smart jammer that targets specific packets and optimizes the reduction in throughput subject to a jamming energy budget. We also present a technique that applies to the case when jamming costs cannot be estimated (Section 4).
- We carefully analyze the weaknesses inherent in the IEEE802.11 MAC and Link layer protocols that allows jammers to be extremely efficient with their jamming.
- We build a testbed comprising of the USRP/GNURadio platform and present a comprehensive experimental evaluation of four RAAs (mentioned above) in the presence of smart jamming. Our experimental results confirm that a smart reactive jammer can maintain links at a low data rate (1 Mbps) at a minimal jamming cost (Jam only $5 - 8$ packets/s) (Section 6).
- Finally, we propose a set of preliminary mitigation mechanisms with their implementation left for future work (Section 7).

[3]Sequence of differences between two successive rate reciprocals.

fter analyzing the costs, we also study the impact of "self-collisions" – collisions arising due to contention among the transmitting stations. The phenomenon where self-collisions cause the network throughput to degrade (or even cause zero throughput – collapse) is called *congestion collapse* [16]. We show that in some cases, the adversary can take advantage of self-collisions and smart-jam to trigger congestion collapse, which in turn significantly reduces the Phase-II maintenance cost.

5.1 Initial Cost

For the initial phase cost analysis, we assume that the adversary's goal is to quickly bring down the rates on the victim links. Thus, the general strategy would be to intensively jam all the packets whose rates are higher than the target rate. We analyze the initial cost for the three specific RAAs as follows.

ONOE: To make ONOE decrease the current rate to the next lower rate, the average number of transmissions per packet (including retransmissions) needs to be more than 1 during a monitor window (1 second). Therefore, in order to decrease the rate by L levels, the adversary has to jam an average of one transmission per packet for each of the L windows. Hence, the cost would be L packets within a monitor window. Note that ONOE counts the total number of retries, thus jamming the retransmissions is also effective.

AMRR: AMRR decreases the rate when it encounters two consecutive failures. Thus, the initial cost for AMRR to decrease the rate by L levels is $2 \cdot L$.

SampleRate: The analysis for SampleRate is more complicated as its behavior is less predictable compared to ONOE and AMRR. When decreasing the rate, SampleRate does not always decrease to the next lower rate. Here, we estimate an upper bound on the initial cost. SampleRate has a black-list policy that any rates with more than 3 consecutive failures will not be considered as a candidate for a period of 2 seconds. Therefore, an effective way for the initial phase is to black-list all the rates higher than the target rates. Assuming there are L' rates higher than the target rate, the initial cost for SampleRate is at most $4 \cdot L'$.

5.2 Maintenance Cost

In general, the value of t depends on the type and value of jammer budget. In Section 6, we will discuss budget types when describing the evaluation metrics. For now, we assume that t is infinite.

ONOE: In the case of ONOE, the low data rate can be maintained by keeping the credit value constant. During a monitor window, if we jam 10% of the packets, the credit will be decreased by 1. Then in the next window, we can jam only one packet (no failure will directly trigger the rate increase), and let the credit increase back to the previous value. By repeating this process, we can prevent ONOE from ever increasing the credit value to 10, thus keeping the rate unchanged. Assuming during each window the sender sends roughly the same amount of packets, the maintenance cost for ONOE is to jam 5% of the packets.

AMRR: In AMRR, the rate is increased after 10 consecutive successes. Thus, jamming one packet after 9 successes will ensure that AMRR continues to use the same rate. So the maintenance cost for AMRR is to jam 10% of the packets.

SampleRate: SampleRate probes other potentially-better rates after every 10 packets and update the statistics for that rate. If we jam all the probes, the transmitting rate stays the same. Therefore, the maintenance cost for SampleRate is also to jam 10% of the packets.

We verify these values later in the evaluation section.

5.3 Impact of Self Collisions on Maintenance Cost

The WLAN's regular traffic also generates self-collision among all the contending stations. These self collisions may keep some RAAs from increasing the transmission rate, especially when traffic demands are high. This would lead to a significant reduction in maintenance cost for the jammer, specifically in some cases, we will see that this may even lead to a zero maintenance cost for the adversary (the jammer triggering the congestion collapse).

In this subsection, we take ONOE as a case study to analyze the impact of the self collisions under saturated traffic conditions. We use PI to indicate the probability of interference caused by self collisions. According to Bianchi's model [4], we can estimate PI given the number of contenting stations, n. Here, we omit the details and list the calculated PI for some values of n in the following table.

n	2	3	5	10
PI	0.057	0.1044	0.1779	0.2894

Recall that ONOE maintains a history of packet successes and failures within a monitor window of 1 second. It increases the credit by 1 if more than 90% of packets succeed; otherwise it decreases the credit by 1. When the credit reaches 10, the sender increases the transmission rate to the next higher rate. Suppose that during a window of a second, the sender has transmitted s packets. The number of successes in this period is a random variable X that follows a binomial distribution $X \sim B(s, 1-PI)$. The probability of increasing the credit is thus $P_{incC} = Pr(X \geq 0.9 \times s)$.

Now, we calculate the probability P_{incR} that the rate increases. There are two events that can cause an increase in data rate: all transmissions succeed during a window, or the credit reaches 10. For a particular window, let us denote the probability of all successes as P^s_{suc}. In practice, s is sufficiently large. Hence, P^s_{suc} is close to 0. Therefore, we focus on calculating P_{credit} defined as the probability that the credit reaches 10. Let $P(t,i)$ be the probability that after t monitor windows, the credit value is $i(\forall i < 10)$. Initially, $P(0,0) = 1$. For $i \leq 8$,

$$P(t,i) = P(t-1,i-1) \times P_{incC} + P(t-1,i+1) \times (1-P_{incC})$$

For $i = 9$,

$$P(t,i) = P(t-1,i-1) \times P_{incC}$$

The probability of increasing the rate by the credit value reaching 10 after t windows is,

$$P_{credit}(t) = P(t,10) = P(t-1,9) \times P_{incC}$$

Thus, $P_{incR} \simeq P_{credit}(t) = P(t-1,9) \times P_{incC}$.

Using the values of PI from the table above, we can see that ONOE is highly vulnerable to self collision. Even with only 3 contending nodes in the channel, P_{incR} turns out to be less than 0.1%. Therefore, in a network with size $n \geq 3$, the maintenance cost for ONOE is reduced by at least a factor of 10 due to self collisions.

In summary, this analysis concludes that some RAAs such as ONOE do not perform well under saturated network conditions which allows the adversary to launch more efficient jamming attacks with much smaller maintenance cost.

6. IMPLEMENTATION AND EVALUATION

In this section, we first discuss the details of our novel testbed designed to implement smart jamming attacks against different RAAs: the hardware and software components, test-bed setup, the jammer implementation, evaluation methodology and performance metrics. Then, we evaluate the efficiency of the attacks and validate the claims made in Section 4 and 5 using results from real world experiments.

6.1 Testbed Topology

For our experiments, we only consider a single link network since our earlier study (Figure 2) show that targeting a single victim link for jamming can cause the whole network throughput to degrade effectively with minimal jamming cost (reflection attack).

6.1.1 System Layout

Our testbed consists of two communication nodes, a sender and a receiver, and the adversary as shown in Figure 3. The sender sends UDP traffic to the receiver to saturate the network. The attacker constantly sniffs the channel and when it sees a sender packet destined for the receiver node of interest, it jams the packet before the transmission is over. For this to happen, the jamming packet must overlap with the sender packet at the receiver side as shown in the time line graph (See Figure 3). Figure 5 depicts the actual test-bed setup we use to run experiments.

6.1.2 Basic Hardware and Software Components

The hardware components of our testbed includes two PCs, (A) a sender node and (B) a receiver node, (C) jammer host, (D) jammer radio and RF-cables and splitters/combiners. We chose to use the RF-cabled setup for our experiments because of following two reasons:

- To achieve reproducible results
- To isolate our testbed from the laboratory network (this includes preventing the jammer from affecting lab network)

All of the above would be hard to achieve in an open medium. Note that operating the nodes with antennas in an open medium will only make the jamming more effective because of additional collisions/losses due to the propagation environment and external traffic.

The software components of our testbed include software defined radio (SDR) for signal processing, a traffic generator, a traffic sniffer tool, and the open source wireless card driver. Later, we will see why the use of open source wireless driver is necessary for the implementation of our reactive jammer.

Testbed Hardware Specification: Our jammer radio is a USRP board [38], which consists of a motherboard and two daughter boards. Each of the daughter boards are capable of operating independently as a transceiver. We use the first daughter board to sniff the channel as the jammer's sensing counterpart that triggers the second daughter board used to jam the channel. We chose D-Link WDA-1320 PCI express wireless cards for our experiments. They run on Atheros AR5212 chipsets that are compatible with open source Madwifi driver [36].

Testbed Software Specification: We use open source GNURadio [39] as the Software Defined Radio (SDR) that runs on USRP

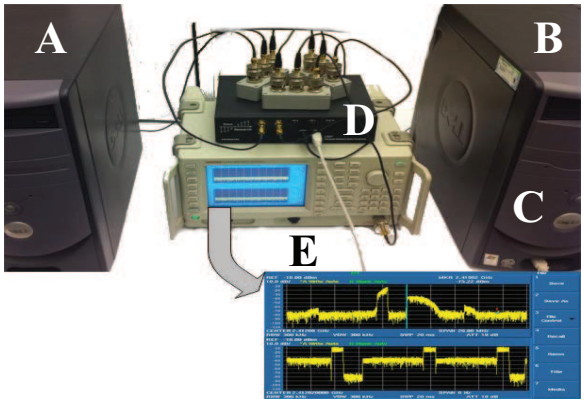


Figure 5: Experimentation Test-bed: (A) Sender, (B) Receiver, (C) Jammer-host, (D) USRP+Splitters+RF-cables, (E) Analyzer.

Component	Version/Model
Host CPUs	Intel Core2 6300
Jammer Radio Motherboard	USRP1
Jammer Radio Daughter boards	RFX-2400
Sender and Receiver Wireless Cards	D-Link WDA-1320 PCI express
Splitter/Combiner	HyperLink SC2402
RF-Cables	L-com RG174 RF-Coaxial Cable

Table 1: Experimental Testbed Hardware Specifications.

to implement fully reactive jammers that are channel aware and can sense and jam the channel within a quick turn around time, if it decides to do so on a per packet basis. Iperf is used as the traffic generator, and Wireshark as the receiver sniffer/analyzer in our testbed.

Component	Version/Model
Host OS	Ubuntu v9.10
Sender Traffic Generator	Iperf v2.0.4
Receiver Sniffer/Analyzer	Wireshark v1.2.7
Jammer SDR	GNURadio v3.3.0
Sender and Receiver Wireless Driver	Madwifi v0.9.4

Table 2: Experimental Testbed Software Specifications.

6.1.3 *Types of Jammers*

In our experiments, we consider the following four kinds of jammers:

- **Continuous Jammer:** This jammer produces a continuous signal at a specified power level. We use this kind of jammer to introduce channel noise into our testbed.
- **Periodic Jammer:** This jammer produces a periodic pulse of fixed size enough to destroy a packet if hit. The idle interval is the input to this kind of jammer and is based on the jammer budget as well as the desired network throughput.
- **Memoryless Jammer:** This jammer is similar to the periodic jammer, except the length of the period is decided using a memoryless distribution, the mean of which is the input parameter for the jammer.
- **Reactive Jammer:** This jammer is channel aware and jams reactively using the information it decodes from the IEEE802.11 PLCP header. The implementation of this kind of jammer requires more explanation, the details of which is provided below.

6.1.4 *Implementation of the Reactive Jammer*

The reactive jammer has two counterparts as shown in Figure 3. The main goal of this jammer is to be able to sniff all the packets in the medium (carried out by C_1 counterpart), and jam the packets destined for the receiver node of interest using an optimal jamming strategy (carried out by the C_2 counterpart). The sniffer's job is to sniff all the packets in the channel, decode only the PLCP IEEE802.11 header, which is always sent at the robust rate of 1.0 Mbps (we disable short preambles), and make jamming decisions on a per-packet-basis using the jamming algorithm described in Section 4. The ultimate goal for all the jammers is to keep the victim link at the the lowest data rate possible in the most efficient way which would result in an overall total network throughput reduction (reflection attack) and may even cause congestion collapse making the optimal use of its jamming budget. This jammer is the best of all the jammers described above.

Limitations: USRP has an inherent hardware limitation that only allows USRP to sample at most 8MHz band. This is a problem because IEEE802.11 communication uses 20MHz band. Furthermore, USRP uses USB to communicate with the host, which has a bandwidth limitation of 32MB/sec. This causes a delay in the order of milliseconds between sensing the channel, passing the information up to the host, host making the decision and asking the USRP to send a jamming signal into the channel [38, 39]. This is ultimately the bottleneck in our testbed implementation of a fully reactive jammer.

To mitigate these limitations, we apply following remedies to our experimental setup:

Remedy: First, with the choice of our hardware (USRP), the jammer can only samples 8MHz band out of 20MHz band of WiFi communication, in turn giving up on the quality of the received samples. However, once the preamble (sent at 1 Mbps) is detected, the jammer only have to decode the 802.11 PLCP header to extract rate information needed for the jamming algorithms discussed in Section 4. Note that this does not allow the jammer to differentiate between the receivers (MAC address is not known from the PLCP header). But, for the case of a single link network that we consider in our experiment, this suffices. If we were to run experiments with reactive jamming in a multiple links scenario, we must improve on our testbed hardware to be able to sample a larger band, and decode more information off the packet (such as receiver MAC address) to target the victim node for jamming.

With the setup shown in Figure 5, we can achieve the reaction time (including the turnaround from sensing to jamming plus the USB delay) of around 2ms. Obviously, this is not enough to be

able to jam higher rate packets, even if we set the packet size to be 1470 bytes. To alleviate this issue, we modify the Madwifi driver to reduce 20MHz bandwidth of IEEE802.11 transmission to 5MHz. This makes the data packet transmission four times longer than when sent using the normal 20MHz band. This allows us as the jammer to jam high data-rate within our testbed. Note that this modification while allowing to complete our experiments does not impact the IEEE802.11 MAC and RAA behavior. In our future work, we plan on using better hardware that will make this problem go away so that we do not need the narrow band remedy.

6.1.5 Assumptions

We make following assumptions in our experimentation:

1. We use RF-cables and splitters/combiners in our experimentation setup. We use continuous jammers to induce noise into our emulation of a wireless channel. This type of setup is typical for evaluating wireless communication systems and achieve reproducible results using channel emulator [40].

2. We consider a single link scenario for the evaluation of the reactive jammer. We focus on jamming a single node with the idea that this would trigger the reflection attack on the network with multiple links, thus optimizing the use of jamming resources. In Figure 2, we run simple experiments in a multiple link scenario, and show that reflection attack can be easily executed.

6.2 Evaluation

In this section, we evaluate the actual performance of reactive jamming against various RAAs. We also compare them with the performance of an oblivious jammer against the same set of RAAs within the same experimentation setup. We first describe the experimentation methodology, the metrics used to evaluate the performance, and then present the experimentation results at the end.

Experimentation Setup: We run our experiments in an RF-cabled setup as described above and depicted in Figure 5. This allowed us to isolate our testbed from the surrounding interference, hence, we were able to achieve results that show very little variance. Each experiment runs for a specified set of parameters is defined by the sender continuously sending saturated traffic for 50 second period. We repeat each experiment 10 times to eliminate the margin of error, which is already very small for us. Since, the retransmissions from the sender are sent using MRR implemented at the driver level (thus, not captured by the Wireshark running at the sender side), we split the sender RF output and connect it to an extra node that sniffs all the retransmissions and logs all the rates used and number of packets sent (retransmissions included) for analysis.

Parameters: The set of parameters used for experimental evaluation is provided below:

Parameter	Setting
Packet Size	1470 bytes
Frequency	2.462 GHz (Channel 11)
Traffic Type	UDP
Traffic Bandwidth	1MB
Noise Power	-20 dB

Table 3: Parameter Specification

Performance Metrics: In Section 5, we describe Phase-I and Phase-II costs as the initial and maintenance phase costs for the jammer to trigger RAA to drop the data rate to the lowest level and keep it there for a certain epoch t. Time parameter t of the maintenance phase depends on the jammer budget. There are two possible scenarios for the jammer budget assigned to the jammer, one that sets the budget as the rate (X Joules per second), and another that assigns total energy (Y Joules) to be used regardless of the time period. Phase-I depends on this definition of the budget. If time is the constraint, it make sense for the jammer to blindly jam all packets and bring down the data rate abruptly, otherwise, it can optimize the jamming energy used when there is no constraint on time spent in achieving the Phase-I goal. This is why, we evaluate the Phase-I cost in both scenarios (See Figure 6). Note that jamming all packets may trigger easier jammer detection than jamming only the non-1 Mbps data traffic.

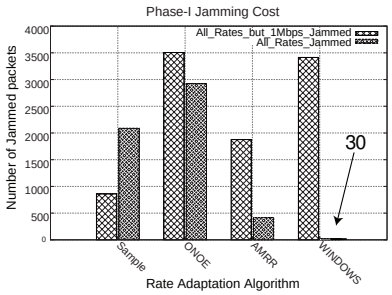


Figure 6: Phase I cost for reactive jammer with (a) not allowing any traffic to go through (b) allowing only 1Mbps traffic to go through, for four different RAAs

In the following, we measure the performance of the jammer in terms of packets jammed during Phase-I and Phase-II.

Evaluation Results:

1. Phase-I cost: This is the cost for the reactive jammer to bring down the link to the lowest data rate (1Mbps). Figure 6 illustrates the cost for the two scenarios described above, one that allows only 1Mbps to go through, and another that jams all data-rate packets. As we can see, for the SampleRate, jamming all but 1Mbps packets performs better than jamming all the packets. This can be explained by the fact that in SampleRate, the multi rate retry parameters $(r_0, r_1, r_2, r_3) = (r, 1\text{Mbps}, 1\text{Mbps}, 1\text{Mbps})$, and the total number of tries is 8 per packet. So, if the jammer jams all the packets, it has to jam 9 packets per transmission. However, if the jammer jams only non-1Mbps packets, it won't have to jam so many packets before SampleRate decides that 1Mbps data packets are the only ones succeeding. For ONOE, it does not seem to differ much either way, and for AMRR jamming all packets performs better than jamming only the non-1Mbps packets. These can be explained by the fact that ONOE counts retransmission failures the same way as it counts the original transmission failure, and the MRR count for AMRR (4 retransmissions) is much smaller than SampleRate and ONOE. For windows RAA, it seems that jamming all packets abruptly brings the link down to 1Mbps.

2. Phase-II cost: This is the cost to maintain the link at 1Mbps. Figure 7 illustrates Phase-II cost for the reactive jammer to maintain RAAs at 1Mbps. This effectively supports our claim that smart jammers can efficiently keep the network at a low throughput with minimal jamming. As we can see, jamming 8, 5, and 6 packets per second can cause SampleRate, ONOE, and AMRR to maintain the network at the lowest data-rate. We do not show the histogram for Windows RAA

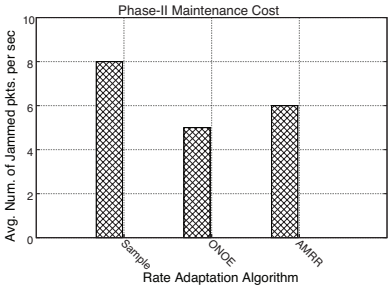


Figure 7: Phase II maintenance cost for reactive jammer for three different RAAs

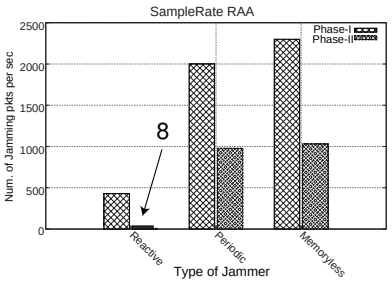


Figure 8: Phase I and Phase II cost comparision for three different kinds of jammers

because, we observed that it is impossible to maintain Windows RAA at 1Mbps. Even if the 2Mbps packet transmission continuously fails, it attempts sending packet at 2Mbps after each 1Mbps successful attempt.

3. Cost comparison among different types of jammers: Finally, we compare the efficiency of reactive jammer in terms of Phase-I and Phase-II costs against oblivious jammers. Figure 8 illustrates the comparison between the performance of periodic jammer, memoryless jammer, and the reactive jammer for the case of SampleRate. As we can see, reactive jammer needs only half the energy that the periodic or the memoryless jammer requires to achieve the same end-goal.

To this end, we have demonstrated that existing rate adaptation algorithms for IEEE802.11 are highly vulnerable to smart jamming attacks, which can result in a significant degradation of the network capacity at an extremely low jamming cost for the adversary.

7. PRELIMINARY MITIGATION TECHNIQUES

In the following, we sketch several mitigation techniques that can prevent smart jamming by severely limiting the amount of key information that can be inferred by an adversary. This lack of information then forces the adversary to operate as a memoryless jammer. In the future, we plan to design full fledged mitigation mechanisms, analyze them, and carry a detailed evaluation of their performance.

Concealing explicit and implicit rate information: The rate information can be protected using post-coding encryption. The encryption should not conflict with the decoding process to preserve the properties of the error correction code. This can be achieved by generating a cryptographic stream based on a shared secret key and a random initialization vector. The initialization vector is sent in the clear as the first sequence following the frame preamble and before the PLCP header. The PLCP header and the packet are XORed with the cryptographic stream. The secret key should either be pre-shared or established using an appropriate authentication and key establishment protocol [13, 24, 31]. This technique will allow to protect the SIGNAL field from being eavesdropped by an adversary and also from guessing the implicit error correction code being used.

To further protect against implicit rate guessing using modulation constellation analysis, we propose to use the same high-order QAM and provide robustness through a larger set of coding rates. This can be achieved with Trellis Coded Modulation [34].

Finally, IEEE802.11g and IEEE802.11a should not be mixed with IEEE802.11b as it would be easy to distinguish between the two physical layers (i.e., OFDM vs. DSSS/CCK).

An adversary might still be able to guess partial information about the rate from the duration of a transmission. This can be protected by using constant duration transmissions through packet length adjustments. However, this is not necessary in practice, since when monitoring the duration, the adversary only knows if a higher rate or a lower rate is being used at the end of the transmission. For example, an adversary can only guess if the rate is 54Mbps by examining the duration of the packet when the transmission is over and whence too late to jam the high rate packets.

Unpredictable rate selection rules: While for a protocol such as ONOE, it might be hard to make it resilient against smart-jamming because of its highly predictable rate selection rules, the popular SampleRate protocol can easily be protected through randomized probing. Instead of sending a probe every ten packets, the probing order should be randomized, furthermore the probed rates should not be sequential but randomly selected unlike the Madwifi implementation.

Interference differentiation: Differentiation between malicious and non-malicious interference, in general, is a difficult problem. Some mechanisms can be used to detect the presence of a reactive jammer. For example, interrupting the transmission for a short period of time within the packet (the location is cryptographically derived) or placing a training sequence at a cryptographic location within the packet allowing the receiver to detect if a jamming signal is present.

8. CONCLUSION AND FUTURE WORK

In this paper, we investigated the robustness of IEEE802.11 RAAs against smart jamming attacks. We consider several classes of RAAs, and specially analyze three state-of-the-art RAAs: SampleRate, ONOE, and AMRR. We evaluate these three RAAs and Windows RAA using our carefully designed GNURadio/USRP aided testbed. We present optimal jamming strategies that exploit the weaknesses found in IEEE802.11 MAC and RAAs. Finally, we discuss the mitigation techniques to alleviate the vulnerabilities that allows the jammer to execute very smart and efficient jamming attacks. In the future, we plan on implementing those mitigation techniques and carry a detailed evaluation of their performance. Our future work also includes studying performance of more current RAAs against smart jamming attacks.

9. REFERENCES

[1] B. Awerbuch, A. Richa, and C. Scheideler. A jamming-resistant mac protocol for single-hop wireless networks. In *ACM PODC*, 2008.

[2] E. Bayraktaroglu, C. King, X. Liu, G. Noubir, R. Rajaraman, and B. Thapa. On the performance of IEEE 802.11 under jamming. In *Proceedings of IEEE INFOCOM*, 2008.

[3] M. A. Bender, M. Farach-Colton, S. He, B. C. Kuszmaul, and C. E. Leiserson. Adversarial contention resolution for simple channels. In *SPAA*, 2005.

[4] G. Bianchi. Performance analysis of the IEEE 802.11 distributed coordination function. *IEEE Journal on Selected Areas in Communications*, 2000.

[5] J. Bicket. Bit-rate selection in wireless networks. *MIT Master's Thesis*, 2005.

[6] I. Broustis, K. Pelechrinis, D. Syrivelis, S. V. Krishnamurthy, and L. Tassiulas. Fiji: Fighting implicit jamming in 802.11 wlans. *SecureComm*, 2009.

[7] J. Camp and E. Knightly. Modulation rate adaptation in urban and vehicular environments: Cross-layer implementation and experimental evaluation. *MobiCom*, 2008.

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

[9] C. C. Chen, H. Luo, E. Seo, N. Vaidya, and X. Wang. Rate-adaptive framing for interfered wireless networks. In *INFOCOM*, 2007.

[10] J. Chiang and Y.-C. Hu. Cross-layer jamming detection and mitigation in wireless broadcast networks. In *MobiCom*, 2007.

[11] M. Heusse, F. Rousseau, G. Berger-Sabbatel, and A. Duda. Performance anamoly of 802.11b. *INFOCOM*, 2003.

[12] G. Holland, N. Vaidya, and V. Bahl. A rate-adaptive mac protocol for multihop wireless networks. *ACM MOBICOM*, 2001.

[13] T. Jin, G. Noubir, and B. Thapa. Zero pre-shared secret key establishment in the presence of jammers. In *MobiHoc*, 2009.

[14] G. Judd, X. Wang, and P. Steenkiste. Efficient channel-aware rate adaptation in dynamic environments. *MobiSys*, 2008.

[15] A. Kamerman and L. Monteband. Wavelan ii: A high-performance wireless lan for the unlicensed band. *Bell Labs Technical Journal*, 1997.

[16] G. Keiser. *Local Area Networks*. McGraw-Hill, 2002.

[17] J. Kim, S. Kim, S. Choi, and D. Qiao. Cara: Collision-aware rate adaptation for IEEE 802.11 wlans. *IEEE INFOCOM*, 2006.

[18] N. Koci and M. K. Marina. Understanding the role of multi-rate retry mechanism for effective rate control in 802.11 wireless lans. *IEEE Conferene on Local Computer Networks*, 2009.

[19] M. Lacage, M. H. Manshaei, and T. Turletti. IEEE 802.11 rate adaptation: A practical approach. *ACM MSWiM*, 2004.

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

[21] G. Lin and G. Noubir. On link layer denial of service in data wireless lans. *Wirel. Commun. Mob. Comput.*, 5(3):273–284, 2005.

[22] G. Noubir and G. Lin. Low-power dos attacks in data wireless lans and countermeasures. *ACM SIGMOBILE*, 7(3):29–30, 2003.

[23] K. Pelechrinis, S. V. Krishnamurthy, C. Gkantsidis, and I. Broustis. Ares: An anti-jamming reinforcement system for 802.11 networks. *CoNEXT*, 2009.

[24] C. Pöpper, M. Strasser, and S. Čapkun. Anti-jamming broadcast communication using uncoordinated spread spectrum techniques. *IEEE J.Sel. A. Commun.*, 28(5):703–715, 2010.

[25] H. Rahul, F. Edalat, D. Katabi, and C. Sodini. Frequency-aware rate adaptation and mac protocols. *MobiCom*, 2009.

[26] K. Ramachandran, R. Kokku, H. Zhang, and M. Gruteser. Symphony: Synchronous two-phase rate power control in 802.11 wlans. *MobiSys*, 2008.

[27] K. B. Rasmussen, S. Capkun, and M. Cagalj. Secnav: secure broadcast localization and time synchronization in wireless networks. In *MobiCom*, 2007.

[28] S. Rayanchu, A. Mishra, D. Agrawal, S. Saha, and S. Banerjee. Diagnosing wireless packet losses in 802.11: Separating collision from weak signal. In *INFOCOM*, 2008.

[29] B. Sadeghi, V. Kanodia, A. Sabharwal, and E. Knightly. Opportunistic media access for multirate ad hoc networks. *ACM MOBICOM*, 2002.

[30] M. K. Simon, J. K. Omura, R. A. Scholtz, and B. K. Levitt. *Spread spectrum communications; vols. 1-3*. Computer Science Press, Inc., NY, 1986.

[31] M. Strasser, C. Popper, S. Capkun, and M. Cagalj. Jamming-resistant key establishment using uncoordinated frequency hopping. In *ISSP*, 2008.

[32] P. Tague, M. Li, and R. Poovendran. Probabilistic mitigation of control channel jamming via random key distribution. In *Proceedings of International Symposium on Personal, Indoor and Mobile Radio Communications*, 2007.

[33] P. Tague, D. Slater, G. Noubir, and R. Poovendran. Linear programming models for jamming attacks on network traffic flows. In *WiOpt*, 2008.

[34] G. Ungerboeck. Channel coding with multilevel/phase signals. *IEEE Transactions on Information Theory*, pages 55–67, 1982.

[35] `http://madwifi-project.org/wiki/UserDocs/RateControl`. Rate control in madwifi.

[36] `http://madwifi.org/`. Multiband atheros driver for wifi.

[37] `http://madwifi.org/browser/trunk/ath_rate/onoe`. Onoe rate control.

[38] `https://www.ettus.com`. Universal software radio peripheral.

[39] `http://www.gnuradio.org`. Gnuradio.

[40] `http://www.home.agilent.com/agilent/product.jspx?pn=11759C`. Agilent technologies rf channel simulator.

[41] M. Vutukuru, H. Balakrishnan, and K. Jamieson. Cross-layer wireless bit rate adaptation. *SIGCOMM*, 2009.

[42] S. H. Wong, H. Yang, S. Lu, and V. Bharghavan. Robute rate adaptation for 802.11 wireless networks. *ACM MobiCom*, 2006.

[43] W. Xu, K. Ma, W. Trappe, and Y. Zhang. Jamming sensor networks: attack and defense strategies. *IEEE Network*, 20(3):41–47, 2006.