Robust Detection of MAC Layer Denial-of-Service Attacks in CSMA/CA Wireless Networks

Alberto Lopez Toledo, *Student Member, IEEE*, and Xiaodong Wang, *Fellow, IEEE*

***Abstract*—Carrier-sensing multiple-access with collision avoidance (CSMA/CA)-based networks, such as those using the IEEE 802.11 distributed coordination function protocol, have experienced widespread deployment due to their ease of implementation. The terminals accessing these networks are not owned or controlled by the network operators (such as in the case of cellular networks) and, thus, terminals may not abide by the protocol rules in order to gain unfair access to the network (selfish misbehavior), or simply to disturb the network operations (denial-of-service attack). This paper presents a robust nonparametric detection mechanism for the CSMA/CA media-access control layer denial-of-service attacks that does not require any modification to the existing protocols. This technique, based on the M-truncated sequential Kolmogorov–Smirnov statistics, monitors the successful transmissions and the collisions of the terminals in the network, and determines how "explainable" the collisions are given for such observations. We show that the distribution of the explainability of the collisions is very sensitive to changes in the network, even with a changing number of competing terminals, making it an excellent candidate to serve as a jamming attack indicator. Ns-2 simulation results show that the proposed method has a very short detection latency and high detection accuracy.**

***Index Terms*—Carrier-sensing multiple-access with collision avoidance (CSMA/CA), denial-of-service (DoS) attack, IEEE 802.11, Kolmogorov–Smirnov (KS), media-access control (MAC), sequential detection.**

I. Introduction

THE carrier-sensing multiple-access with collision avoidance (CSMA/CA) protocol relies on the random deferment of packet transmissions for contention resolution and efficient use of the shared channel among nodes in a network [1]. Being a completely distributed algorithm, its correct operation assumes that all nodes obey the protocol. However, wireless devices can easily modify their software parameters to gain unfair access to the network (selfish misbehavior), or simply to prevent other nodes from accessing it (denial-of-service (DoS) attack).

Manuscript received September 10, 2007; revised April 29, 2008. Published August 13, 2008 (projected). This work was supported in part by the U.S. National Science Foundation (NSF) under Grant CMS 0427353 and in part by the U.S. Office of Naval Research (ONR) under Grant ONR N00014-03-1-0039. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Christina Fragouli.

A. L. Toledo is with Telefonica Research, Barcelona 08021, Spain (e-mail: alopezt@tid.es).

X. Wang is with the Department of Electrical Engineering, Columbia University, New York, NY 10027 USA (e-mail: wangx@ee.columbia.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIFS.2008.926098

Selfish misbehavior has been analyzed in the past [29], usually under a game theoretic framework [2]–[4]. Selfish terminals are necessarily exposed, as their objective is to increase their own transmissions in the network, which makes their traffic susceptible to statistical analysis and identification [1], [5]. Media-access control (MAC) DoS attacks, on the other hand, are stealthy by nature, as the attacker does not have to reveal itself in order to perform the attack. More important, MAC DoS attacks require very little power [6] as only specific portions of the other terminal's transmissions need to be targeted in order to succeed. Such jammers, which jam the network with the knowledge of the target MAC protocol, are known as protocol-aware jammers [7]. MAC DoS attacks through intelligent jamming are, as a result, much easier and more efficient to perform, and their impact on the network performance is often catastrophic: an energy-efficient stealthy jammer can disrupt selected control packets and reduce the network throughput to zero. On the other hand, the random operation of the CSMA/CA protocol, together with the nature of the wireless medium itself, makes network conditions appear different for different terminals [1]. Hence, it is difficult to determine whether errors are caused by a surge in the number of legitimate terminals (such as in hotspots), or by malicious terminals. For simplicity, we assume that all transmission errors are caused by simultaneous transmissions of other terminals (legitimate or misbehaving), so in effect, all errors are caused by collisions. Our model can easily be extended to consider other error causes, such as poor channel conditions.

While research on DoS attacks to 802.11 networks is active and prolific, most works focus primarily on attacks related to user authentication and network disconnection [8], [9], rather than on throughput-reduction attacks [10]. And of those concerned with throughput, they mainly focus on physical RF jamming in order to provoke bandwidth exhaustion [11], [12]. In this paper, we deal with MAC vulnerabilities [13] and, in particular, those related to intelligent jammers (i.e., jammers that operate using knowledge of the MAC layer protocol). Some partial solutions have been proposed in the literature but they have been proved unsuccessful so far. In particular, [14] and [15] propose modifications to the IEEE 802.11 distributed coordination function (DCF) that account for an increase in the number of collisions in the network. While such approaches can detect some attacks, the modification of the protocol requires an update of the IEEE 802.11 installed base, making it difficult to deploy. Other related works, such as [6], [7], [16], and [17], rather than trying to identify attacks, explore methods to perform more efficient attacks. We will use some of these efficient attacks as benchmarks for performance evaluation of our detector.

In order to perform the detection of a jamming attack,[1] we first show that it is possible to determine the probability that a terminal is contributing to an observed collision by tracking its successful transmissions. We then introduce the concept of explainability of a collision (i.e., the probability that a collision can be explained by the events observed in the network). We show that the distribution of the explainability of the collisions is very sensitive to jamming attacks. Finally, we propose detecting a jamming attack by detecting the event that the distribution of the explainability of the collisions deviates significantly from that under normal operating conditions, using a robust nonparametric Kolmogorov–Smirnov detector.

The remainder of this paper is organized as follows. Section II describes the CSMA/CA protocol and its vulnerability to DoS attacks, and formulates the detection problem. In Section III, we characterize the operation of the IEEE 802.11 DCF protocol under "normal" operation and introduce the concept of explainability of collisions. We propose our MAC DoS detection algorithm in Section IV. Ns-2 simulation results are given in Section V. Finally, Section VI concludes this paper.

II. CSMA/CA Protocols and Problem Formulation

While the techniques presented in this paper apply to any CSMA/CA protocol, we will focus our attention, without loss of generality, on the IEEE 802.11 DCF protocol.

A. IEEE 802.11 DCF

The IEEE 802.11 DCF protocol is a CSMA/CA protocol that defines two distinct techniques to access the medium: 1) the basic access and 2) the RTS/CTS access [18].

In the basic access, the terminals implement a two-way handshake mechanism. A terminal senses the channel to be idle before starting a transmission. If the channel is idle, then the terminal is allowed to transmit. If during this sensing time, the channel appears to be busy at any time, the terminal defers the transmission and enters into the collision avoidance (CA) mode. In CA mode, the terminal generates a random backoff interval during which it waits before attempting another transmission. This random backoff is used to minimize the probability of collision between terminals accessing the medium. The idle time is slotted, and the terminals are allowed to transmit only at the beginning of the slot time.

The random backoff timer is uniformly chosen between $[0, v)$, where v is called the contention window and satisfies $v \in [\mathrm{CW}_{\min}, \mathrm{CW}_{\max}]$, where $\mathrm{CW}_{\min}$ and $\mathrm{CW}_{\max}$ are called the minimum and maximum contention windows, respectively. At the first transmission attempt, v is set to $\mathrm{CW}_{\min}$. The backoff timer is decremented while the channel is idle (i.e., it only counts the idle time). If at any time the channel is sensed to be busy, the backoff timer is paused until the channel is sensed to be idle again. When the backoff timer reaches 0, the terminal then transmits. Following the successful reception of the data, the receiving terminal transmits an ACK to the transmitting terminal. Upon reception of the ACK, the backoff stage is reset to 0 and $v = \mathrm{CW}_{\min}$. If the source terminal does not receive the ACK after a timeout period (ACK_timeout) or it detects the transmission of any other frame in the channel (collision), the frame is assumed to be lost. After each unsuccessful transmission, the value of v is doubled up to a maximum of $\mathrm{CW}_{\max} = 2^m \mathrm{CW}_{\min}$, where m is usually referred to as the maximum backoff stage [19]. In the sequel, we will consider the standard IEEE 802.11 DCF with $\mathrm{CW}_{\min} = 32$ and $\mathrm{CW}_{\max} = 1024$.

The RTS/CTS access is similar to the basic access, but it makes use of a four-way handshake protocol in which prior to data transmission, a terminal transmits a special short request-to-send frame (RTS) to reserve the transmission and reduce the cost of collisions.

B. Vulnerability of IEEE 802.11 DCF to DoS Attacks

An intelligent jammer transmits with the knowledge of the protocol [7], distinguishing between the control packets and the data packets by analyzing the length of packets and the interpacket timings. In particular, an attacker can easily corrupt those frames for which their exact transmission times are known: CTS/RTS frames, ACK frames, and DATA frames [6]. An attacker only needs a small portion of the energy of a packet to corrupt it. Unfortunately, the repercussion of the attacker is not limited to the lost frame. Due to the distributed operation of the CSMA/CA protocols, a node being jammed will defer the transmission of its next frame following the multiplicative decrease algorithm. A terminal undergoing a few successive jams would virtually stop transmission. This is a strong incentive not only for malicious attacks, but for selfish misbehaving nodes that can "clear" the network for its own legitimate transmissions, without having to reveal their wrongdoing.

To illustrate the effect that an intelligent attack has on IEEE 802.11 DCF, let us consider an IEEE 802.11 DCF network with 15 terminals where only one external terminal jams the CTS frames. We consider two types of jamming: 1) random uniform jamming in which the jammer corrupts every CTS frame with certain probability and 2) ON–OFF jamming, in which an attacker starts jamming with certain probability, always corrupting the next five CTS frames. Fig. 1 is obtained by using the ns-2 simulator and shows the normalized throughput of the jammed network compared to the energy spent by the jammer, also measured in throughput (i.e., the amount of time it spends corrupting frames). It is seen that a jammer has an incredible impact on the network utilization, virtually driving throughput to zero while using an energy of three orders of magnitude less than that of a legitimate terminal.

C. Problem Formulation

While preventing jamming attacks is not possible, the detection of such attacks is of paramount importance. Upon detection of a DoS attack, the network should have a mechanism to inform the terminals and the neighboring access points about the threat in order to take appropriate actions. For example, the compromised access point may be turned off to prevent other user from logging. In the case of multihop systems, such as wireless mesh networks, the detection of jamming attacks is important in order to reroute the traffic avoiding the compromised areas. The specifics of such mechanisms, however, fall outside the scope of this paper.

[1]For the rest of the text, we would refer to "jamming" as MAC jamming attacks.

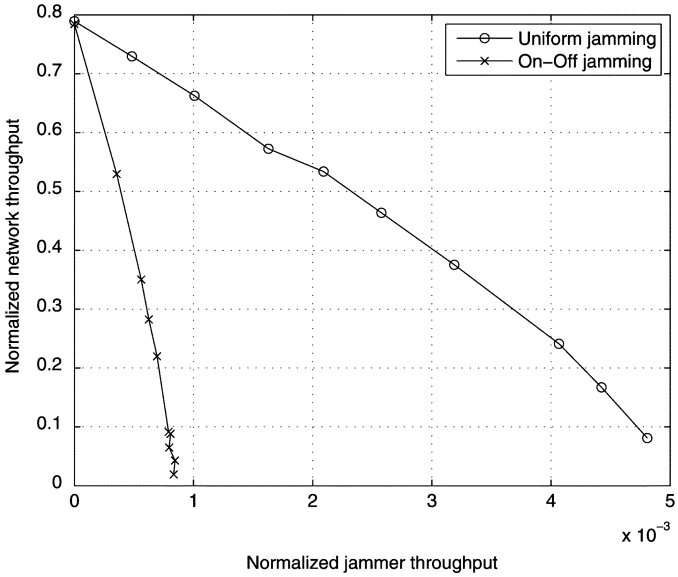


Fig. 1. Throughput of an IEEE 802.11 DCF in the presence of just one "intelligent jammer."

Obviously, a jamming attack will result in an increase in the number of collisions observed in the network. However, because, in general, it is not possible to identify specific collisions caused by a jammer, our approach is to observe the variability in the distribution of the collision, differentiating between "normal" and "abnormal" (jamming) operation. Specifically, let $y_1, \ldots, y_K$ be a sequence of observations related to the operation of a CSMA/CA network (we discuss the specific observations we propose in Section III). We consider two hypotheses: 1) hypothesis H_0 corresponds to the network performing normally (i.e., no jamming attack) and 2) hypothesis H_1 corresponds to the case where the network is jammed. Hence, we have the following hypothesis test, as shown in (1) at the bottom of the page, where f_0 and f_1 are the probability distributions of the observations when the network is operating normally and when the network is being jammed, respectively.

Note that there may be other causes of transmission errors (i.e., collisions) that are not necessarily caused by jamming, such as poor channel conditions or the presence of hidden terminals. However, the occurrence of such events can be easily factored into f_0 by, respectively, estimating the average number of transmission errors for a given received signal-to-noise ratio (SNR), and estimating the average impact of hidden terminals in collisions.

III. Statistical Analysis of Collisions Under Normal Operation

From the point of view of an IEEE 802.11 DCF terminal, time can be slotted into variable length slots. Specifically, one time slot will either correspond to a fixed length idle slot, a transmission slot, or a collision slot. Transmission and collision slots have variable length depending on the length of the packets being transmitted and whether the RTS/CTS mode is enabled.

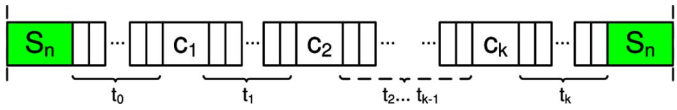


Fig. 2. Sequence of states in the network between two consecutive successful transmissions S_n of terminal n.

One logical choice of the observations $\{y_1, \ldots, y_K\}$ in (1) is the collision probability observed in the network. However, the collision probability in IEEE 802.11 DCF is a function of the number of competing terminals,[2] and hence a change in the collision probability may be caused by a change in the number of terminals in the network, and not (necessarily) by a jamming attack. Moreover, and as we will see in Section IV, the collision probability is not very sensitive to a jamming attack, especially for a high number of competing terminals. Hence, the collision probability may not be a good indicator for a jamming attack. In what follows, we present an alternative characterization of the "normal" operation of an IEEE 802.11 DCF network that 1) does not depend directly on the observed collision probability and 2) is more sensitive to the presence of jammers and, hence, offers improved detection capabilities.

A. Contribution of a Terminal to a Collision

Consider an IEEE 802.11 DCF network with N competing terminals as described in Section II-A, and consider the state of the network between two consecutive successful transmissions of terminal n. Since the backoff timers are decremented only during idle slots and not during transmissions of other terminals, without loss of generality, we can ignore the transmission slots of terminals other than n. Then, the sequence of states in the network between two consecutive successful transmissions S_n of terminal n will have the form depicted in Fig. 2. Assume that K collisions occur between the two transmissions of terminal n, denoted by $\{c_1, \ldots, c_K\}$, and define the corresponding idle slot sequence as $\{t_0, \ldots, t_K\}$ (i.e., there are t_i idle slots between the consecutive collisions c_{i-1} and c_i). We want to determine the probability that terminal n has participated in each collision c_i based on $\{t_0, \ldots, t_K\}$.

Define a binary random variable x_i^n with $x_i^n = 1$ if terminal n contributed to collision c_i, and $x_i^n = 0$ otherwise. We call the sequence $\{x_1^n, \ldots, x_K^n\}$ a collision codeword. We are interested in calculating the probabilities $p(x_i^n = 1 \,|\, t_0, \ldots, t_K), i = 1, \ldots, K, n = 1, \ldots, N$, where

[2]The number of competing terminals is the number of terminals that have something to send at a particular time, not the total number of terminals registered, for example, to an access point.

$$\text{choose} \begin{cases} H_0 : y_1, \ldots, y_K \sim f_0 & \text{(normal operation)} \\ H_1 : y_1, \ldots, y_K \sim f_1 & \text{(abnormal operation–i.e., jamming)} \end{cases} \tag{1}$$

$p(\cdot|\cdot)$ denotes the conditional probability. Denote W_i and w_i as the window size and the backoff counter[3] of the terminal after collision c_i, respectively. Initially, after a successful transmission, the terminal n has a window size of $W_0 = 32$ and a backoff counter of $w_0 = 0$. The terminal randomly selects the backoff time of $u \sim \mathcal{U}[0, 32]$ before attempting the next transmission. Then, the probability that the terminal contributed to collision c_1 is

$$p(x_1^n = 1 \,|\, t_0) = P(u \le t_0) = t_0/W_0. \tag{2}$$

If $x_1^n = 1$, the terminal n would increase the window size to $W_1 = 64$ and would again select a backoff time of $u \sim \mathcal{U}[0, 64]$. On the other hand, if $x_1^n = 0$, the terminal n does not change the backoff window size, so $W_1 = 32$, and sets $w_1 = t_0$. By the memoryless property of the uniform distribution, if $p(w) = \mathcal{U}[0, W]$, then we have $p(w \,|\, w > t) = \mathcal{U}[0, W - t]$. So, if $x_1^n = 0$, it is equivalent to terminal n randomly picking a new backoff time $u \sim \mathcal{U}[0, 32 - t_0]$. We see that the state of the terminal after the collision c_i depends only on the state of the terminal after the collision slot c_i and the number of idle slots t_{i-1} between the collisions c_{i-1} and c_i.

Generalizing, after collision c_{i-1}, let the current window size of terminal n be W_{i-1} and let the backoff counter w_{i-1} be the number of idle slots since the last transmission attempt (i.e., the number of idle slots since the last event $x_j^n = 1, j < i$). We say that the state of terminal n is (W_{i-1}, w_{i-1}). Then, at collision c_i, we have

$$p(x_i^n = 1 \,|\, W_{i-1}, w_{i-1}, t_{i-1}) = \frac{t_{i-1}}{W_{i-1} - w_{i-1}}. \tag{3}$$

When $x_i^n = 1$, terminal n would update its state to $(W_i, w_i) = (\min(2W_{i-1}, W_{\max}), 0)$, where $W_{\max}$ is the maximum window size allowed by the protocol (e.g., $W_{\max} = 1024$ in the standard IEEE 802.11). On the other hand, if $x_i^n = 0$, terminal n would update its state to $(W_i, w_i) = (W_{i-1}, w_{i-1} + t_i)$. Finally, after the last collision c_k, the probability of the terminal having a successful transmission after t_K idle slots is given by

$$p\left(x_{K+1}^n = 1 \,|\, W_K, w_K, t_K\right) = \frac{t_K}{W_k - w_k}. \tag{4}$$

Note that this is equivalent to having a trailing bit $x_{K+1}^n = 1$ in the collision codeword.

We can use the aforementioned probabilities to construct a trellis similar to that of a linear block code. To illustrate the construction of the trellis, consider the example in Fig. 3 that shows the state of the network between two transmissions of terminal 1. We observe two collisions $\{c_1, c_2\}$ and the idle slot sequence $\{t_0, t_1, t_2\} = \{1, 5, 8\}$. Also note that there are two transmissions from other terminals denoted as B (or busy slots) that are ignored for the purpose of terminal 1's trellis construction. After the first successful transmission, the state of terminal 1 is $(W_0, w_0) = (32, 0)$. Then, $p(x_1^1 = 1 \,|\, t_0 = 1) = 1/32$. If $x_1^1 = 1$, then the terminal would double its maximum window size $W_1 = 64$ and set $w_1 = 0$. On the other hand, if $x_1^1 = 0$, then the terminal would keep $W_1 = 32$ and set $w_1 = 1$. This would continue until the next successful transmission is encountered. Note that the trellis can be pruned if a certain branch has probability 0; for example, if $t_i > (W_{i-1} - w_{i-1})$.

[3]The backoff counter keeps track of the number of idle slots since the last attempted transmission.

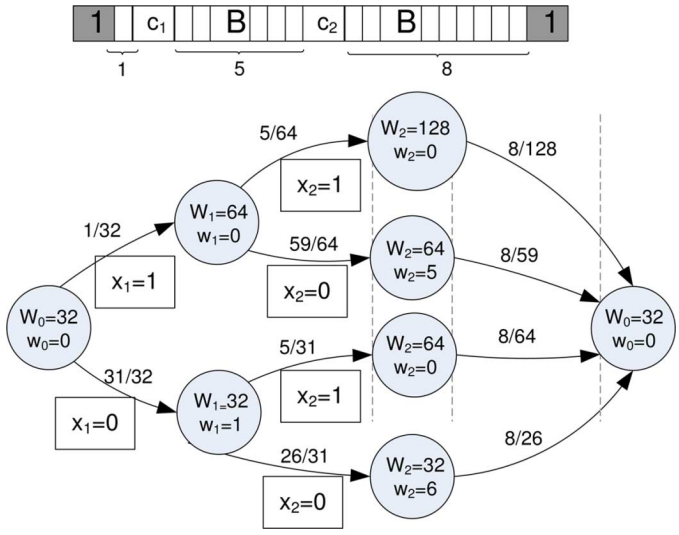


Fig. 3. Trellis corresponding to the idle slot sequence $\{t_0, t_1, t_2\} = \{1, 5, 8\}$.

From the trellis that is shown, we can compute the probability of each codeword for terminal n by multiplying the transition probabilities along the corresponding path in the trellis. For example, in Fig. 3, we can calculate the probability of the collision codeword $\{x_1^1, x_2^1\} = \{1, 0\}$ as

$$\begin{aligned} &p\left(\{x_1^1, x_2^1\} = \{1,0\} \,|\, \{t_0, t_1, t_2\} = \{1, 5, 8\}\right) \\ &\quad = p\left(x_1^1 = 1 \,|\, W_0 = 32, w_0 = 0, t_0 = 1\right) \\ &\qquad \times p\left(x_2^1 = 0 \,|\, W_1 = 64, w_1 = 0, t_1 = 5\right) \\ &\qquad \times p\left(x_3^1 = 1 \,|\, W_2 = 64, w_2 = 5, t_2 = 8\right) \\ &\quad = (1/32) \times (59/64) \times (8/59). \end{aligned} \tag{5}$$

Finally, we can estimate the probability of individual bits by marginalization as

$$\begin{aligned} &p(x_i^n = 1 \,|\, t_0, \ldots, t_K) \\ &\qquad = \sum_{x_i^n = 1} p(x_1^n, \ldots, x_K^n \,|\, t_0, \ldots, t_K). \end{aligned} \tag{6}$$

The computation of the marginal probabilities in (6) can be efficiently implemented by using the forward–backward algorithm [20].

Algorithm 1: Monte Carlo approximation of the marginal probabilities in (6)

1) Let $\{t_0, \ldots, t_K\}$ be the sequence of idle slots since the last transmission of n (see Fig. 2). Let M denote the number of Monte Carlo iterations.

2) Denote by M_i the number of occurrences of $x_i^n = 1$. Set $M_i = 0, \quad i = 1, \ldots, K$.

3) Denote $L_{\text{valid}} = 0$ as the number of valid samples.

4) **for** $l = 1, \ldots, L$ **do**

5) Set $W_0 = 32, w_0 = 0$.

6) Set $x_i^n = 0, i = 1, \ldots, K+1$.

7) **for** $i = 1, \ldots, K+1$ **do**

8) **if** $t_i > (W_{i-1} - w_{i-1})$ **then**

9) This codeword is not feasible. Drop the sample.

10) **break**

11) **end if**

12) **if** $\text{rand}() \leq t_i/(W_{i-1} - w_{i-1})$ **then**

13) $x_i^n = 1$.

14) $W_i = \max(2W_{i-1}, W_{\max})$. $w_i = 0$.

15) **else**

16) $x_i^n = 0$.

17) $W_i = W_{i-1}$. $w_i = w_{i-1} + t_i$.

18) **end if**

19) **end for**

20) **if** $x_{K+1}^n \neq 1$ **then**

21) This codeword is not feasible. Drop the sample.

22) **else**

23) $M_i = M_i + x_i^n, \forall i = 1, \ldots, K$.

24) Set $L_{\text{valid}} = L_{\text{valid}} + 1$.

25) **end if**

26) **end for**

27) Finally, the value of (6) can be obtained by

$$p(x_i^n = 1 \,|\, t_0, \ldots, t_K) = \frac{M_i}{L_{\text{valid}}}.$$

Approximation of $p(x_i^n \,|\, t_0, \ldots, t_K)$: Note that the total number of nodes at the Kth stage of the trellis is, in the worst case, 2^K. However, in "normal" operation of IEEE 802.11 DCF, and with a number of competing terminals ranging from 1 to 25, we observe from ns-2 simulations that the chance of having $K > 20$ is only about 0.9%. As an indication, the forward–backward algorithm for (6) takes less than 3 s in decoding codewords with length 20, and a few milliseconds for decoding codewords of length 15 and lower using a standard desktop PC. However, when a jammer is present, the codeword lengths increase and the exact computation of (6) becomes intractable. In those cases, a simple Monte Carlo approximation can be used, as shown in Algorithm 1.

Also, it is important to note that the Hamming weight of the codewords $\{x_1^n, \ldots, x_K^n\}$ (i.e., the number of x_i^n terms that are equal to 1) is proportional to $1/(1 - p_c)$, where p_c is the probability of collision [e.g., estimated as (11)] (i.e., the codewords are sparse). We can exploit this fact when using the Monte Carlo approximation by discarding codewords whose Hamming weights are much larger than $1/(1-p_c)$. Finally, if the codeword length K is much larger than 20, it is safe to ignore the codeword because it is very unlikely that the collisions were produced in a "normal" operation of the network. This simplification, while favoring the case of a jammer being present, does not impose a great penalty because such a large codeword is a very rare event, and is indeed most likely caused by "abnormal" conditions in the network.

Validating $p(x_i^n \,|\, t_0, \ldots, t_K)$: We can use the values $p(x_i^n \,|\, t_0, \ldots, t_K)$ in (6) to estimate the other properties of an IEEE 802.11 DCF network. Consider a network with N competing terminals. We observe the network for a certain period of time and observe C collisions $\{c_1, \ldots, c_C\}$. Denote $x_i = \sum_{n=1}^{N} x_i^n$ as the actual number of terminals that participate in the collision c_i. Then, the collision factor, defined as the average number of terminals that participate in the collisions in the network [1], is given by $\Gamma = (1/C)\sum_{i=1}^{C} x_i$. While the values $\{x_i^n\}$ and, hence, the collision factor Γ can only be directly measured in simulations, in a real network, only the idle slot sequence $\{t_0, \ldots, t_K\}$ is relevant.[4] to collision c_i can be measured. Then, to estimate Γ, we estimate $p(x_i^n \,|\, t_0, \ldots, t_K)$ given in (6), and use the following estimator:

$$\hat{\Gamma} = \frac{1}{C} \sum_{i=1}^{C} \sum_{n=1}^{N} p(x_i^n \,|\, t_0, \ldots, t_K). \tag{7}$$

Fig. 4 shows the estimation of the collision factor given in (7) compared to that obtained in ns-2 using the simulation parameters described in Section V-A. We can see that when the number of competing terminals is small, there is some uncertainty about the contribution of terminals to collisions. However, as the number of terminals in the network increases, the estimate $\hat{\Gamma}$ perfectly matches the exact value. With Γ being a physical verifiable property of the network, the accuracy of the estimation confirms that our analytical formulation of $p(x_i^n \,|\, t_0, \ldots, t_K)$ is sound.

Some Notes on $p(x_i^n \,|\, t_0, \ldots, t_k)$: The quantity $p(x_i^n \,|\, t_0, \ldots, t_K)$ depends only on the number of idle slots between successful transmission $(t_0, \ldots, t_K)$ of terminal n. It is important to note that the values of $(t_0, \ldots, t_K)$ are observable by any terminal in range without collaboration from other stations.[5] Moreover, the values of $(t_0, \ldots, t_K)$ do not depend on the transmissions nor the state of the measuring station and, hence, can be calculated, for example, by a station that never transmits. This property allows the introduction of specific-purpose "black-box" detectors, without modification of existing terminals or access points, allowing the introduction of the DoS detector in existing IEEE 802.11 deployments.

[4]Denote S_i^n and S_{i+1}^n as two consecutive successful transmissions of terminal n, and let $\{c_j, c_{j+1}, \ldots, c_{j+p}\}$ and $\{t_h, \ldots, t_{h+p+1}\}$ be the sequence of collisions and the sequence of idle slots in the network, respectively, between transmissions S_i^n and S_{i+1}^n. We say that $\{t_h, \ldots, t_{h+p+1}\}$ is the sequence of idle slots that are relevant to collisions $\{c_j, c_{j+1}, \ldots, c_{j+p}\}$. While each collision in the network may have a different relevant idle sequence, we will generically refer to it as $\{t_0, \ldots, t_K\}$.

[5]See [1] for a detailed discussion as to how the values of $\{t_0, \ldots, t_K\}$ can be collected in an 802.11 network.

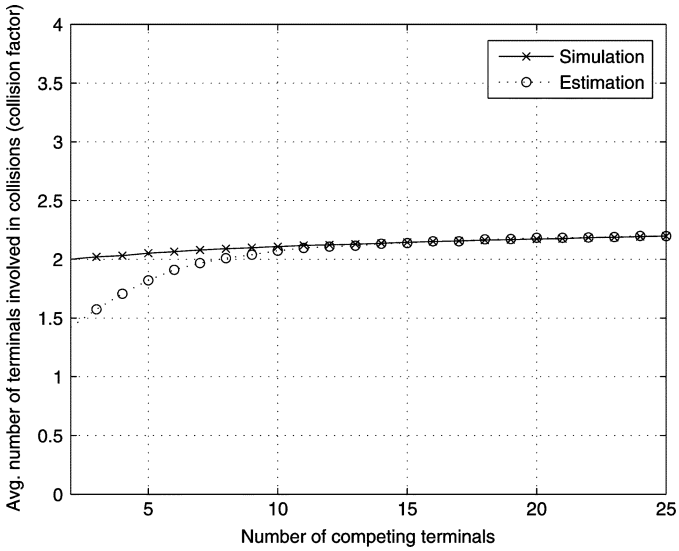


Fig. 4. Estimation of the collision factor Γ via (7).

B. Explainability of Collisions

Next, we consider the combined contribution of all the terminals in the network to a collision c_i, and we will use it to determine whether the collision is a legitimate one or it is abnormal, such as in the case of a jamming attack.

Consider an IEEE 802.11 DCF network with N competing terminals, and let $\{c_1, c_2, \ldots, c_C\}$ be the sequence of collisions in the network over a period of time. We can use (6) to calculate the probability $p(x_i^n \,|\, t_0, \ldots, t_K)$ that each terminal n contributed to each collision c_i, where $\{t_0, \ldots, t_K\}$ refers to the idle slot sequence relevant to collision c_i. When there is no jammer present in the network, a collision event is defined as

$$c_i = \mathbb{1}\left\{\sum_{i=1}^{N} x_i^n \geq 2\right\} \tag{8}$$

where $\mathbb{1}(\cdot)$ is the indicator function. In practice, since $\{x_i^n\}$ are not observable, we define the explainability of collision c_i as

$$\begin{aligned} e(c_i) &\triangleq \mathbb{E}\left\{\mathbb{1}\left\{\sum_{i=1}^{N} x_i^n \geq 2.\right\}\middle|\, t_0, \ldots, t_K\right\} \\ &= P\left(\sum_{n=1}^{N} x_i^n \geq 2 \middle|\, t_0, \ldots, t_K\right) \\ &= 1 - \prod_{n=1}^{N} p\left(x_i^n = 0 \,|\, t_0, \ldots, t_K\right) \\ &\quad - \sum_{n=1}^{N} p\left(x_i^n = 1 \,|\, t_0, \ldots, t_K\right) \\ &\quad \times \prod_{j \neq n} p\left(x_i^j = 0 \,|\, t_0, \ldots, t_K\right) \end{aligned} \tag{9}$$

(i.e., the probability that there are at least two terminals contributing to collision c_i).

To illustrate the procedure to calculate $e(c_i)$, consider the sequence of network states depicted in Fig. 5. For each terminal n in the network, we calculate the probabilities $p(x_i^n \,|\, t_0, \ldots, t_K)$ that the terminal had participated in collision c_i. In order to calculate these probabilities, we use either the forward–backward algorithm or Algorithm 1 between each consecutive pair of successful transmissions of terminal n. This calculation is simultaneously performed for all N terminals competing in the network. Then, for each collision c_i for which all of the probabilities $p(x_i^n \,|\, t_0, \ldots, t_K), n = 1, \ldots, N$ are known, we use (9) to calculate its explainability $e(c_i)$. The process is detailed in Algorithm 2. Note that the algorithm proceeds sequentially, updating the values of the probability that a terminal n contributed to the collisions observed in the network since its last successful transmission. The algorithm cannot calculate $e(c_i)$ until the contributions of all N competing terminals to collision c_i are obtained.

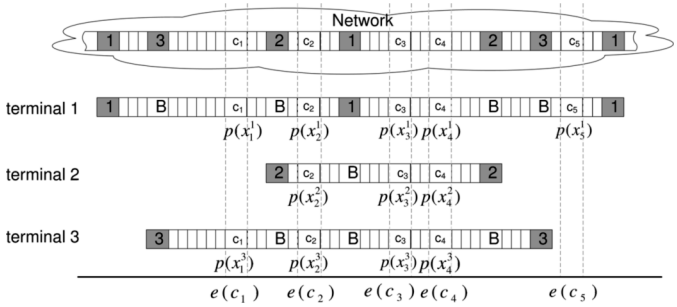


Fig. 5. Example of calculating $e(c_i)$. The conditionals on the relevant idle slot sequences $\{t_0, \ldots, t_K\}$ are not shown for brevity.

Algorithm 2: Calculation of collision explainability

1) Observe the network until a successful transmission from any terminal is observed. Let that be a transmission from terminal $n \in \{1, \ldots, N\}$.

2) Denote $\{c_1, c_2, \ldots, c_C\}$ as the sequence of collisions and $\{t_0, \ldots, t_K\}$ as the sequence of idle slots observed in the network since the last successful transmission of terminal n.

3) Use (6) to calculate the contribution of n to the collisions $\{c_1, c_2, \ldots, c_C\}$ (i.e., $p(x_1^n \,|\, t_0, \ldots, t_K), \ldots, p(x_C^n \,|\, t_0, \ldots, t_K)$).

4) For those collisions $c_i \in \{c_1, c_2, \ldots, c_C\}$ for which all of the $p(x_i^j \,|\, \cdot), j = 1, \ldots, N$ are known (where N is the number of competing terminals in the network), calculate $e(c_i)$ given by (9).

5) Go to Step 1).

Fig. 6(a) and (b) shows the cumulative distribution functions (cdf) of $e(c_i)$ obtained by Algorithm 2 in ns-2 using the simulation parameters described in Section V. Note that as the number of competing terminals in the network increases, the percentage of collisions that can be explained simply by observing the sequence of idle slots in the network increases, and for $N > 8$, virtually all of the collisions can be explained with a probability of at least 0.5. Also, for $N < 8$, there is a significant percentage of collisions that are perfectly explainable (i.e., $e(c_i) = 1$) and, hence, there is a jump in the cdf at $e(c_i) = 1$. The increasing

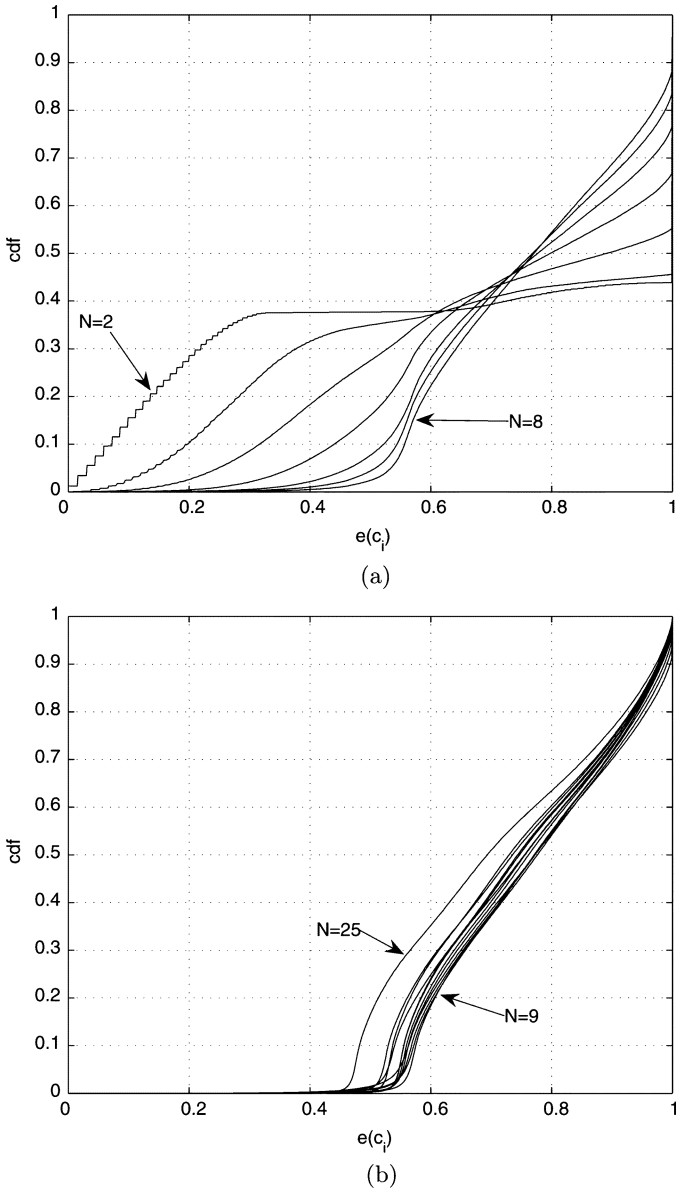


Fig. 6. Distribution of the explainability of the collisions $e(c_i)$ in IEEE 802.11 DCF calculated using Algorithm 2. (a) For the number of competing terminals between 2 and 8. (b) For the number of competing terminals between 9 and 20.

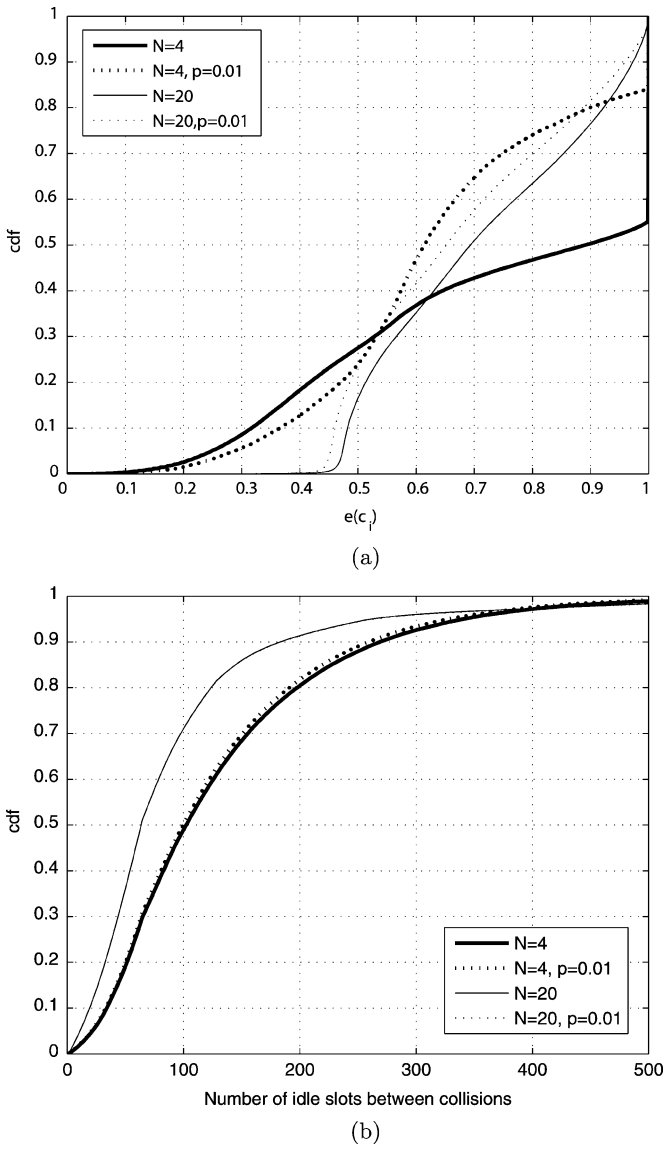


Fig. 7. Effect of a jammer in the network. (a) cdf of collision explainability $e(c_i)$. (b) cdf of the number of idle slots between collisions.

accuracy in the prediction of the collisions accounts for the results obtained in Fig. 4, where we see that accurate estimations can be obtained by using (6). This is a remarkable result, considering that the actual transmissions of the terminals (i.e., the values of x_i^n) are not known.

Sensitivity of $e(c_i)$ to Jamming Attacks: Ideally, if the transmission times of all terminals x_i^n were known, the quantities $e(c_i)$ in (9) would suffice to determine whether a collision is caused by a jammer. On the other hand and under normal protocol operation, the explainabilities $e(c_i)$ have the distribution shown in Fig. 6(a) and (b), and some collisions can be explained better than others. More important, the distribution of $e(c_i)$ is an excellent indicator of the normal protocol operation for a given number of competing terminals. Fig. 7(a), obtained in ns-2 using the simulation parameters specified in Section V, shows the change in the cdf of $e(c_i)$ for a number of competing terminals $N = 4$ and $N = 20$, when there is no jammer in the network, and when there is an attacker present in the network jamming randomly as few as 1% of the frames. There are two important aspects to note: 1) the cdf of $e(c_i)$ may significantly change its shape in the presence of a jammer and 2) the cdf of the explainability of collisions changes dramatically even for a very small percentage of corrupted frames as we can see in Fig. 7(a), for the case of a jammer corrupting only 1% of the CTS frames. This contrasts with the effect that jamming has on the distribution of the frequency of the collisions that as we can see in Fig. 7(b), does not experience a significant change under light jamming, and is almost unnoticeable for the case of $N = 20$. Hence, the quantities $e(c_i)$ are excellent candidates to serve as jamming attack indicators.

IV. MAC DoS Detector

In the previous section, we have shown that the distribution of $e(c_i)$ can be used as observations [i.e., y_i in (1)] for the de-

tection of a jamming attack. However, Algorithm 2 assumes that the number N of competing terminals in the network is known, which is generally not the case. In this section, we first show how to keep track of the number of competing terminals, and then we show how to implement a robust MAC DoS detector.

A. Tracking the Number of Competing Terminals

The problem of estimating the number of competing terminals (i.e., the number of terminal that have something to send) in an IEEE 802.11 DCF network has already been studied [21], [22]. Previous works base their estimations on the assumption that there is a functional relationship between the number of competing terminals and the collision probability, as was first shown in [23]. Unfortunately, in the case of a MAC jamming attack, that functional relationship is no longer valid, because a jammer would cause an increase in the number of collisions without affecting the number of competing terminals. Instead, we propose here to estimate N by direct observation of the successful transmissions of the terminals in the network: if a successful transmission of a terminal is observed, the terminal is definitely competing. However, after a successful transmission is observed, the event of the terminal ceasing to compete (i.e., having no more data to send) is not observable. If a certain amount of time has passed since the last successful transmission of a terminal and a new successful transmission has not occurred, it might be because the terminal has indeed emptied its buffers, or because it is experiencing heavy collisions. In what follows, we propose a method to estimate how long it is necessary to wait after observing a successful transmission of a terminal before concluding that the terminal is no longer competing in the network.

Probability That a Terminal is Still Competing: Define a random variable T as the number of idle slots between successful transmissions of a terminal. Let p_c be the probability that the terminal will suffer from a collision if it transmits in the current slot. It is shown in [1] that in an IEEE 802.11 DCF with $\mathrm{CW}_{\min} = 32$ and $\mathrm{CW}_{\max} = 1024$, we have where $I_{\max}$ is the maximum allowable number of collisions, and $Z = \sum_{i=0}^{I_{\max}} p_c^i(1-p_c)$. Let $\hat{p}_c^{(1)}, \ldots, \hat{p}_c^{(q)}$ be a sequence of collision probability estimates obtained by

$$\hat{p}_c = \frac{\#\text{ of collisions}}{\#\text{ of transmissions}}. \tag{11}$$

The pmf $g_{\text{idle}}(t)$ of T can be estimated by estimating $\hat{p}_c$ and the corresponding $\hat{g}_{\text{idle}}$ by using (11) and (10), as shown at the bottom of the page, respectively, over several intervals and then taking the average. The reader can refer to [1] for specific details about the calculation and derivation of (10).

Let T_n be the random variable denoting the number of idle slots since the last successful transmission of terminal n. Then, the probability that terminal n is still competing after t idle slots is given by

$$\begin{aligned} P_{\mathcal{C}}^n(t) &= P(\mathcal{C} \,|\, T_n > t) \\ &= \frac{P(T_n > t \,|\, \mathcal{C})P(\mathcal{C})}{P(T_n > t \,|\, \mathcal{C})P(\mathcal{C}) + P(T_n > t \,|\, \bar{\mathcal{C}})P(\bar{\mathcal{C}})} \end{aligned} \tag{12}$$

where $\mathcal{C}$ is the event that a terminal is still competing, and $\bar{\mathcal{C}}$ is the event that a terminal has ceased to compete. Note that $P(T_n > t \,|\, \mathcal{C}) = 1 - \hat{G}_{\text{idle}}(t)$, where $\hat{G}_{\text{idle}}(t)$ is the cdf of $\hat{g}_{\text{idle}}(t)$. Also note that if terminal n is not competing, then $P(T_n > t \,|\, \bar{\mathcal{C}}) = 1$. The probability $P(\mathcal{C})$ (i.e., the overall probability that after a successful transmission a terminal still has data to send) depends on the movement and the transmission pattern of the terminals for the specific application. Finally, we decide that a terminal has ceased to compete after τ idle slots after its last transmission where τ is such that $\sum_{i=\tau}^{T_{\max}} P_{\mathcal{C}}^n(i) = \eta, \eta$ is the desired false alarm probability, and $T_{\max} = \max(T) = \sum_{i=0}^{4} 2^i \mathrm{CW}_{\min} + \sum_{i=5}^{I_{\max}} \mathrm{CW}_{\max}$.

We assume that the arrival and departure rate of a terminal in the network occurs in a time scale that is orders of magnitude greater than the time between two consecutive successful transmissions of a terminal (e.g., milliseconds in an IEEE 802.11b network). For our problem, we chose a false alarm probability of $\eta = 0.01$, and we also assume that if a terminal has just transmitted, it is very likely that it will still have data to send, and set $P(\mathcal{C}) = 0.99$, which corresponds to a terminal sending on average 100 frames before ceasing to compete. Note that a conservative large prior $P(\mathcal{C})$ does not reduce the accuracy of the estimation as long as the aforementioned assumption holds true, but instead it adds delay to the decision. However, note that decision speed is not a concern because the calculation of the values $p(x_i^n \,|\, t_0, \ldots, t_K)$ is always delayed until the next successful transmission of terminal n is observed. Also, the number of idle slots to wait before making a decision is never greater

$$T \sim \begin{cases} \mathcal{U}[0,32] & \text{w.p.} \quad 1 - p_c/Z \\ \mathcal{U}[0,32] * \mathcal{U}[0,64] & \text{w.p.} \quad p_c(1-p_c)/Z \\ \mathcal{U}[0,32] * \mathcal{U}[0,64] * \mathcal{U}[0,128] & \text{w.p.} \quad p_c^2(1-p_c)/Z \\ \mathcal{U}[0,32] * \mathcal{U}[0,64] * \mathcal{U}[0,128] * \mathcal{U}[0,256] & \text{w.p.} \quad p_c^3(1-p_c)/Z \\ \mathcal{U}[0,32] * \mathcal{U}[0,64] * \mathcal{U}[0,128] * \mathcal{U}[0,256] * \mathcal{U}[0,512] & \text{w.p.} \quad p_c^4(1-p_c)/Z \\ \mathcal{U}[0,32] * \mathcal{U}[0,64] * \mathcal{U}[0,128] * \mathcal{U}[0,256] & \\ \qquad * \mathcal{U}[0,512] * \mathcal{U}[0,1024] & \text{w.p.} \quad p_c^5(1-p_c)/Z \\ \mathcal{U}[0,32] * \mathcal{U}[0,64] + \mathcal{U}[0,128] * \mathcal{U}[0,256] & \\ \qquad * \mathcal{U}[0,512] * \sum_{i=5}^{6} \mathcal{U}[0,1024] & \text{w.p.} \quad p_c^6(1-p_c)/Z \\ \quad \cdots & \cdots \\ \mathcal{U}[0,32] * \mathcal{U}[0,64] * \mathcal{U}[0,128] * \mathcal{U}[0,256] & \\ \qquad * \mathcal{U}[0,512] * \sum_{i=5}^{I_{\max}} \mathcal{U}[0,1024] & \text{w.p.} \quad p_c^{I_{\max}}(1-p_c)/Z \end{cases} \tag{10}$$

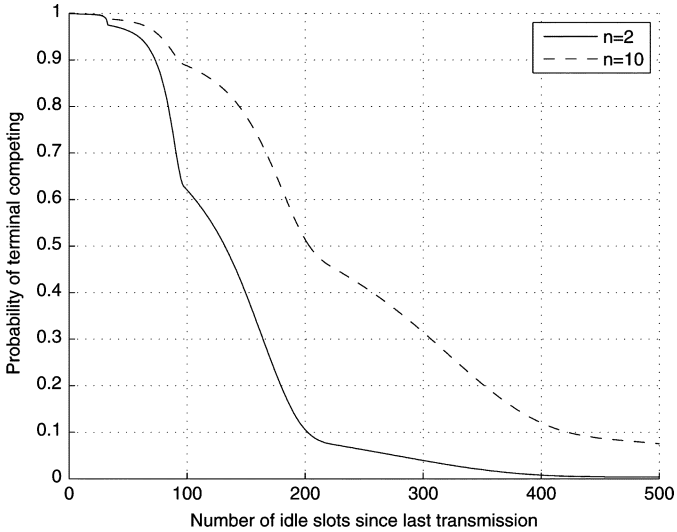


Fig. 8. Probability $P_c^n(t)$ that a terminal is still competing after t idle slots since its last successful transmission.

than $T_{\max}$. Fig. 8 shows an example of (12) for an IEEE 802.11 DCF network with two and ten terminals.

The algorithm to calculate the number of competing terminals for Algorithm 2 proceeds as follows. Denote N' as the total number of terminals that have been observed transmitting in the network in the past, and let T_n be the number of idle slots since the last successful transmission of terminal n. Initially, $T_n = 0, n = 1,\ldots,N'$. Then, for each terminal n and as T_n increases, one of the following two events will occur first: either there is a new successful transmission from terminal n, or $T_n > \tau$, where τ is selected as indicated before. Let $\{c_1,\ldots,c_p\}$ be the sequence of collisions in the network since the last successful transmission of terminal n. If a successful transmission occurs first, then it is obvious that terminal n was competing at collisions $\{c_1,\ldots,c_p\}$. If, on the other hand, we have $T_n > \tau$ first, then terminal n was not competing at collisions $\{c_1,\ldots,c_p\}$. The total number of competing terminals can then be computed by counting the number of terminals that were competing at each collision c_i.

There are two potential sources of error in the aforementioned procedure. First, the algorithm considers that a terminal starts competing (i.e., performs its first transmission in the network) the first time that a successful transmission from that terminal is observed. However, the first transmission of a terminal may not be successful. In fact, a terminal would undergo, on average, $1/(1-p_c)$ collisions before succeeding for the first time. Second, if a terminal that has stopped to compete, starts competing again in less than τ idle slots after its last successful transmission, then the algorithm may erroneously decide that the terminal did not stop competing in the first place. Both type of errors are only relevant if the departure and arrival rate of a terminal are on the same time scale as that of two consecutive transmissions (e.g., milliseconds in an IEEE 802.11b network), which is unlikely. Finally, note that the presence of a jammer does not introduce errors, because the algorithm takes into account the observed probability of collision in the network: in the presence of a jammer, the observed probabilities of collision $\hat{p}_c$ would increase, and so the algorithm would wait for more idle slots before concluding that a legitimate terminal has ceased to compete.

B. Sequential DoS Detector

As we discussed in Section II-C, we are interested in developing a detector that can discriminate between "normal" operation of the network characterized by a probability distribution f_0, and "abnormal" operation characterized by an unknown probability distribution f_1. We will use the sequence of explainability of collisions $e(c_i)$ as our observation variables for the hypothesis testing problem defined in (1), so that $\{e(c_1),\ldots,e(c_K)\} \sim f_i, i = 0,1$. Since the distribution f_1 when a jammer is present is unknown, it is necessary to use a distribution-free or nonparametric approach to perform the detection. Hence, we employ the M-truncated sequential Kolmogorov–Smirnov (KS) test introduced in [1].

The K–S test [24], [25] is the most widely used goodness-of-fit test for continuous data. It is based on the empirical distribution function (edf), which converges uniformly and almost surely to the real population cdf (Glivenko–Cantelli Theorem) [26]. The K–S test is the most powerful two-sample test for which the distribution of the statistic is known, and we use this property to design a sequential test based on it. The K–S test compares the edf F_1 obtained from the data samples with the hypothesized cdf F_0, and determines whether $F_1 = F_0$, $F_1 < F_0$, or $F_1 > F_0$. For the jamming detection problem, we use the following test:

$$\text{choose} \begin{cases} H_0 : F_1 = F_0 & \text{(no jamming)} \\ H_1 : F_1 \neq F_0 & \text{(jamming).} \end{cases} \tag{13}$$

Define F_0^N as the cdf of the sequence of the explainability of collisions in an IEEE 802.11 DCF network with N competing terminals when there is no jammer in the network (Fig. 6). Then, for a given sequence of C collisions in the network $\{c_1,\ldots,c_C\}$, the distribution F_0 for the test in (13) is given by

$$F_0 = \frac{1}{C}\sum_{i=1}^{C} F_0^{N(c_i)} \tag{14}$$

where $N(c_i)$ is the number of competing terminals in the network at collision c_i calculated using the procedure described in Section IV-A. Note that the cdfs $\{F_0^j : j = 0,1,2,\ldots\}$ represent the normal behavior of IEEE 802.11 DCF when no jammer is present, and can be calculated offline via simulations and preloaded in the detector. F_0, on the other hand, depends on the actual number of competing terminals for the observation period, so it has to be calculated online. Also let $\{e(c_1),\ldots,e(c_C)\}$ be the corresponding sequence of the explainability of collisions. Then, the edf F_1 of the observations for the test in (13) is given by

$$F_1(e(c_j)) = \frac{1}{C}\sum_{i=1}^{C} \mathbb{1}\{e(c_i) \leq e(c_j)\}. \tag{15}$$

The K–S test statistic D, defined as the maximum value of the difference between the two cdfs $D \triangleq \max_{-\infty<x<+\infty}\{F_1(x) - F_0(x)\}$, can be calculated as

$$\hat{D} = \max_{1\leq i\leq C}\{F_1(e(c_i)) - F_0(e(c_i))\}. \tag{16}$$

Define

$$\lambda(\hat{D}) = \max\left\{\left(\sqrt{C} + 0.12 + \frac{0.11}{\sqrt{C}}\right)\hat{D}, 0\right\} \quad (17)$$

$$\beta = 1 - \sqrt[M]{1-\alpha} \quad (18)$$

where C is the number of samples (i.e., collisions) and M is the maximum stage of the M-truncated sequential K-S test with a probability of false alarm α [1]. Then, at any stage of the K-S test, the hypothesis H_0 is rejected if $P \leq \beta$, where P is given by [27]

$$P = e^{-2\lambda(\hat{D})^2}. \quad (19)$$

Finally, the algorithm for detecting the presence of a jammer in the network is summarized in Algorithm 3.

Algorithm 3: Detecting jammer attacks using the M-truncated sequential K-S test with $P_{\mathrm{FA}} = \alpha$

1) $m = 0$.

2) $\beta \leftarrow 1 - \sqrt[M]{1-\alpha}$.

3) $m \leftarrow m + 1$.

4) Let $\{e(c_i), \ldots, e(c_{i+C})\}$ be the last values returned by Algorithm 2, and $\{N(c_i), \ldots, N(c_{i+C})\}$ be the number of competing terminals at each c_i calculated as described in Section IV-A.

5) Update F_0 with the new calculated number of competing terminals $\{N(c_i), \ldots, N(c_{i+C})\}$ using (14).

6) Update the edf F_1 with the observations $\{e(c_i), \ldots, e(c_{i+C})\}$ using (15).

7) Calculate the significance level P of the stage as in (19).

8) **if** $P \leq \beta$ **then**

9) reject H_0. The network is behaving "abnormally" (e.g., there is a jammer in the network).

10) **else if** $m = M$ **then**

11) do not reject H_0. The network is behaving normally.

12) **else**

13) go to 2

14) **end if**

V. Simulation Results

A. Simulation Setup

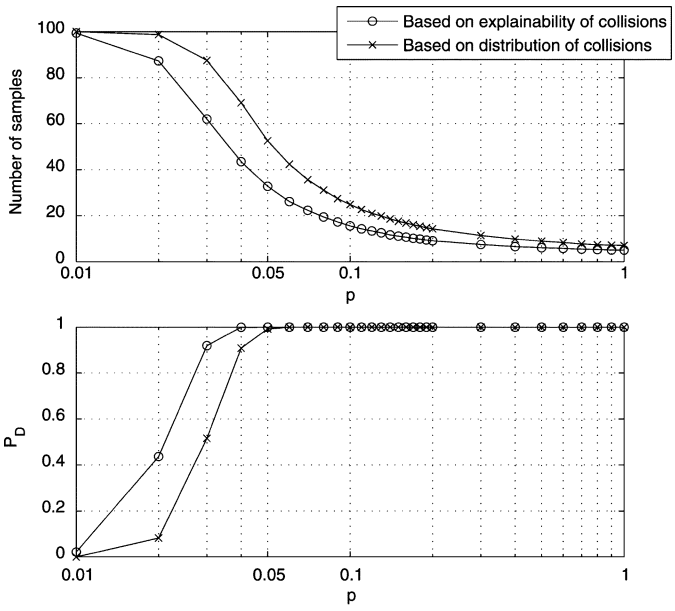
We consider the IEEE 802.11 DCF described in Section II-A where a legitimate terminal uses $\mathrm{CW}_{\min} = 32$ and $\mathrm{CW}_{\max} = 1024$. For all of the experiments in this paper as well as for the figures in the previous sections, data are collected using the ns-2 network simulator version 2.28 [28] and we implement the detection algorithms using MATLAB. We modified the 802.11 implementation so that the nodes measure idle slots in the network and estimate the collision probabilities. The simulated scenario is an IEEE 802.11 network with one access point, and the wireless terminals communicate via UDP with peers outside the wireless network. One terminal (e.g., the access point) monitors the transmissions from all of the other terminals and implements the detection algorithm. The parameters used in the simulation are typical for a 11-Mb/s 802.11b wireless local-area network (WLAN). No packet fragmentation occurs and the nodes are located close to each other to avoid capture or hidden terminal problems. The propagation delay is 1 μs. The packet size is fixed with a payload of 1024 B. The MAC and PHY headers use, respectively, 272 and 192 b. The ACK length is 112 b. The Rx/Tx turnaround time is 20 μs and the busy detect time 29 μs. The short retry limit and long retry limit are set to 7 and 4 retransmissions, respectively. Finally, the slot time is 20 μs, the SIFS is 10 μs, and the DIFS is 50 μs. We model the data arrival as an ON–OFF process, where terminals alternate periods of transmission with periods of silence. Both the ON and the OFF times are modeled after a Pareto(1.5) distribution with burst mean (ON) of 2 s and idle time mean (off) of 5 s.

We only consider the case of intelligent jamming [7] (i.e., the jammer corrupts frames with the knowledge of the IEEE 802.11 DCF protocol). We consider two types of attacks: 1) p-random jamming, where the jammer corrupts the CTS frames in the network with probability p. We will use random jamming with very low p to measure the ability of the detectors to track very small changes in the normal network conditions, for example, in the case of a jammer that tries to avoid detection. A particular case of this attack is full jamming, in which the attacker corrupts every frame in the network ($p = 1$). This attack would correspond to a physical RF jamming attack. The full jamming attack would serve as a benchmark for the speed of the detection of worst-case scenarios and 2) misbehavior jamming, when one of the terminals in the network deliberately selects a different window size from that specified by the protocol in order to gain unfair access to the network. Such a misbehaving node would indeed cause the rest of legitimate nodes to observe more collisions than expected. Note that the algorithm presented here would be able to detect the abnormal behavior in the network but is not designed to identify the misbehaving node.

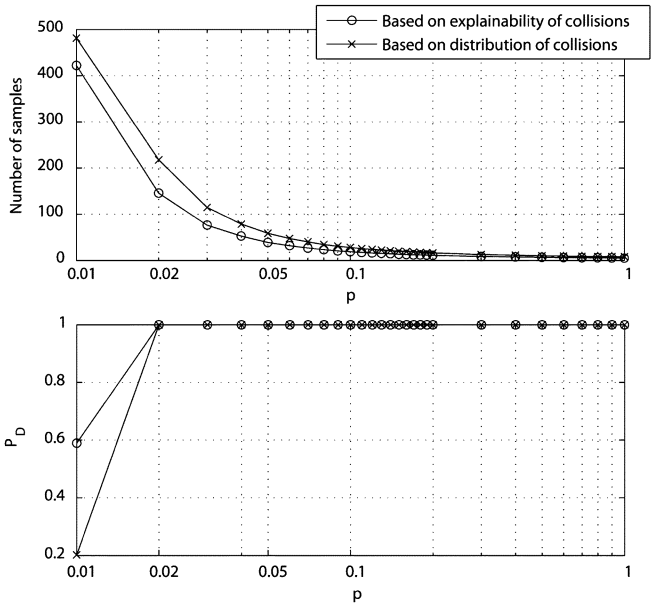
For comparison, apart from the detector described in Algorithm 3, we also consider a similar M-truncated sequential K-S detector that does not use the explainability of collisions as observations, but instead uses the distribution of the number of idle slot between collisions in the network (i.e., how often collisions occur). This comparison would allow us to determine whether a detector based on the explainability of collisions is better suited for this problem than detectors based on the frequency or probability of collisions.

B. Results

Fig. 9 shows the detection performance in the event of a p-random jamming attack, for both the 100- and 500-truncated sequential detector based on the explainability of collisions, and based on the distribution of collisions. As we can see, the performance of the detectors, as expected, is very good for a full jamming attack ($p = 1$), only needing a few samples for the detection. The sequential detector, based on the explainability of collision, consistently outperforms the one based on the distribution of the collisions, needing, on average, a little more than half of the samples. The 100-truncated detector is able to detect the



(a)



(b)

Fig. 9. Number of samples and probability of detection (P_D) of the detectors when a p-random jammer is present and $P_{\rm FA} = 0.01$. (a) 100-truncated K–S. (b) 500-truncated K–S.

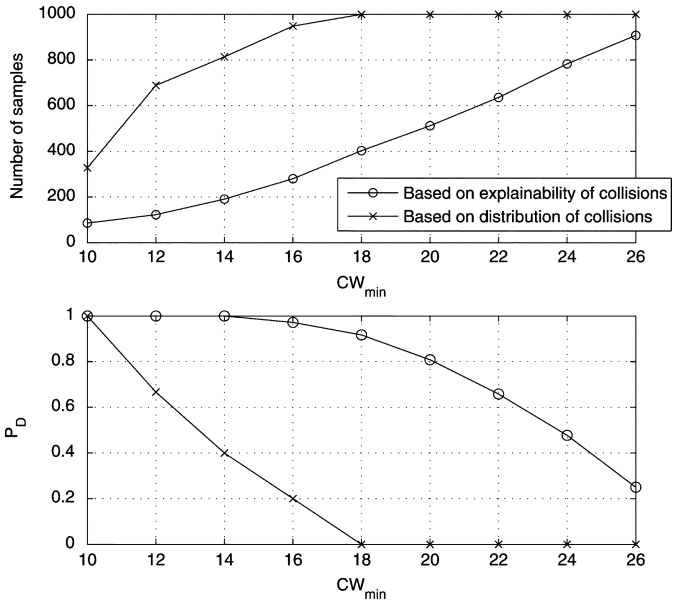


Fig. 10. Number of samples and probability of detection (P_D) of the 1000-truncated detectors when a node is misbehaving and $P_{\rm FA} = 0.05$.

presence of a jammer up to $p = 0.04$ with $P_D \geq 0.95$. In order to detect a 0.02-random attack with $P_D \geq 0.95$, it is necessary to increase the truncation point up to 500. Note that a 0.02-random attack would almost have no effect on the network performance, and still, our detector would identify the attack in less than 100 ms in a standard IEEE 802.11b network, which is remarkable.

The superior performance of the detector based on the explainability of collisions is more evident when the attacks are not uniformly distributed. Fig. 10 shows the performance of the 1000-truncated detectors when there is a misbehaving node in the network that selects different values of $\mathrm{CW_{min}}$ with $\mathrm{CW_{max}} = 2^5\mathrm{CW_{min}}$[1]. Note that the pattern of nonlegitimate collisions caused by a misbehaving node are distributed similar to the transmission pattern of a node (i.e., they follow the CSMA/CA multiplicative increase algorithm). Still, our detector based on the explainability of collisions performs considerably better than the one based on the distribution of collision. We see that our MAC jamming detector is sensitive to any cause of "abnormal" behavior in the network, making it an excellent early-detection intrusion detection system, that cannot only detect the presence of jammers, but also provide input to other detectors, such as the misbehaving detector in [1], to improve their performance.

Finally, Fig. 11 shows the ROC curve of the 500-truncated detectors with $P_{\rm FA} = 0.01$ when a 0.01-random attack is performed. As we can see from the results, the detector based on the explainability of collisions performs better than the one based on the distribution of the collisions. The number of samples for the detection of both random jamming and misbehavior stays in the order of milliseconds for a standard IEEE 802.11b network. Overall, the explainability of the collisions provides superior information about the network conditions and, hence, its use is not limited to the attacks shown here.

VI. Conclusion

We have proposed a method for detecting the MAC layer DoS attacks (i.e., jamming) in a CSMA/CA network, based on calculating the probability that the collisions in the network can be explained by simple observation of the events in the network. The M-truncated sequential K–S test is employed to determine whether the samples are consistent with the hypothesis that the network is operating normally. We apply the test to detect intelligent jamming attacks in an IEEE 802.11 DCF network using the ns-2 simulator. We have shown that the distribution of the explainability of the collisions is an excellent indicator of the presence of jammers and misbehaving nodes in the network, and that it greatly surpasses the standard detectors that track changes in the distribution of the collisions in the network. The proposed technique is robust since it is able to detect any deviation from

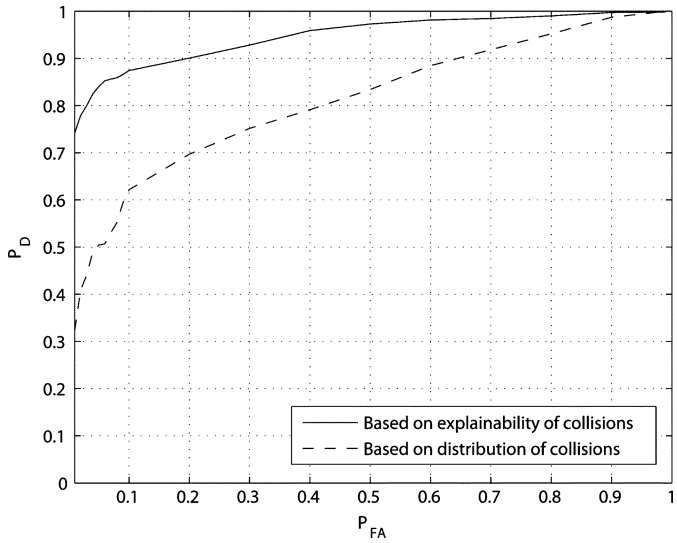


Fig. 11. ROC curve for the 500-truncated detectors when a 0.01-jammer is present.

the "normal" operation of the network, and can operate without modifying the protocol implementation.

REFERENCES

[1] A. L. Toledo and X. Wang, "Robust detection of selfish misbehavior in wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 25, no. 6, pp. 1124–1134, Aug. 2007.

[2] M. Cagalj, S. Ganeriwal, I. Aad, and J.-P. Hubaux, "On selfish behavior in CSMA/CA networks," presented at the IEEE Infocom, Miami, FL, Mar. 2005.

[3] A. MacKenzie and S. Wicker, "Game theory and the design of self-configuring, adaptive wireless networks," *IEEE Commun. Mag.*, vol. 39, no. 11, pp. 126–131, Nov. 2001.

[4] A. MacKenzie and S. Wicker, "Stability of multipacket slotted aloha with selfish users and perfect information," presented at the IEEE Infocom, San Francisco, CA, Apr. 2003.

[5] P. Kyasanur and N. H. Vaidya, "Selfish MAC layer misbehavior in wireless networks," *IEEE Trans. Mobile Comput.*, vol. 4, no. 5, pp. 502–516, Sep./Oct. 2005.

[6] M. Acharya, T. Sharma, D. Thuente, and D. Sizemore, "Intelligent jamming in 802.11b wireless networks," presented at the OPNETWORK, Washington, DC, Aug. 2004.

[7] D. Thuente and M. Acharya, "Intelligent jamming in wireless networks with applcations to 802.11b and other networks," presented at the 2006 IEEE MILCOM, Washington, DC, Oct. 2006.

[8] F. Ferreri, M. Bernaschi, and L. Valcamonici, "Access points vulnerabilities to dos attacks in 802.11 networks," presented at the IEEE Wireless Commun. Net. Conf., Atlanta, GA, Mar. 2004.

[9] L. A. Mohammed and B. Issac, "Dos attacks and defense mechanisms in wireless networks," presented at the Int. Conf. Mobile Tech. Appl. Syst., Guangzhou, China, Nov. 2005.

[10] J. Bellardo and S. Savage, "802.11 denial-of-service attacks: Real vulnerabilities and practical solutions," presented at the USENIX Security Symp., Washington, DC, Aug. 2003.

[11] M. Hall, A. Silvennoinen, and S. G. Haggman, "Effect of pulse jamming on ieee 802.11 wireless lan performance," presented at the IEEE MILCOM, Atlantic City, NJ, Oct. 2005.

[12] W. Xu, W. Trappe, Y. Zhang, and T. Wood, "The feasibility of launching and detecting jamming attacks in wireless networks," presented at the ACM MobiHoc, Urbana-Champaign, IL, May 2005.

[13] A. L. Toledo and X. Wang, "Detecting MAC layer collision abnormalities in CSMA/CA wireless networks," presented at the IEEE Int. Conf. Commun., Beijing, China, May 2008.

[14] R. Negi and A. Rajeswaran, "Dos analysis of reservation based mac protocols," presented at the IEEE Int. Conf. Commun., Seoul, Korea, May 2005.

[15] N. BenAmmar and J. S. Baras, "Incentive compatible medium access control in wireless networks," presented at the Workshop Game Theory Commun. and Net. (GameNets), Pisa, Italy, Oct. 2006.

[16] M. Acharya and D. Thuente, "Intelligent jamming attacks, counterattacks and (counter)^2 attacks in 802.11b wireless networks," presented at the OPNETWORK, Washington, DC, Aug. 2005.

[17] Y. W. Law, L. van Hoesel, J. Doumen, P. Hartel, and P. Havinga, "Energy-efficient link-layer jamming attacks against wireless sensor network mac protocols," presented at the ACM Workshop Security Ad Hoc Sensor Net. (SASN), Alexandria, VA, Nov. 2005.

[18] A. L. Toledo, T. Vercauteren, and X. Wang, "Adaptive optimization of IEEE 802.11 DCF based on bayesian estimation of the number of competing terminals," *IEEE Trans. Mobile Comput.*, vol. 5, no. 9, pp. 1283–1296, Nov. 2006.

[19] C. Wang, B. Li, and L. Li, "A new collision resolution mechanism to enhance the performance of IEEE 802.11 DCF," *IEEE Trans. Veh. Technol.*, vol. 53, no. 4, pp. 1235–1246, Jul. 2004.

[20] D. Mackay, *Information Theory, Inference & Learning Algorithms*. Cambridge, U.K.: Cambridge University Press, 2003.

[21] G. Bianchi and I. Tinnirello, "Kalman fitler estimation of the number of competing terminals in an IEEE 802.11 network," presented at the Infocom, San Francisco, CA, Mar. 2003.

[22] T. Vercauteren, A. L. Toledo, and X. Wang, "Batch and sequential bayesian estimators of the number of active terminals in an IEEE 802.11 network," *IEEE Trans. Signal Process.*, vol. 55, no. 2, pp. 437–450, Feb. 2007.

[23] G. Bianchi, "Performance analysis of the IEEE 802.11 distributed coordination function," *IEEE J. Sel. Areas Commun.*, vol. 18, no. 3, pp. 535–547, Mar. 2000.

[24] F. Massey, "The Kolmogorov-Smirnov test for goodnes of fit," *J. Amer. Statist. Assoc.*, vol. 46, no. 253, pp. 68–78, 1951.

[25] H. Khamis, "The δ-corrected Kolmogorov-Smirnov test for goodness of fit," *J. Statist. Planning Inference*, vol. 24, pp. 317–335, 1990.

[26] H. Khamis, "The two-stage δ-corrected Kolmogorov-Smirnov test," *J. Appl. Statist.*, vol. 27, no. 4, pp. 439–450, 2000.

[27] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery, *Numerical Recipes in C: The Art of Scientific Computing.*. Cambridge, U.K.: Cambridge Univ. Press, 1992.

[28] S. McCanne and S. Floyd, Network simulator 2. [Online]. Available: http://www.isi.edu/nsnam/ns.

[29] S. Radosavac, J. S. Baras, and I. Koutsopoulos, "A framework for MAC protocol misbehavior detection in wireless networks," presented at the ACM Workshop Wireless Security, Cologne, Germany, Sep. 2005.

Alberto Lopez Toledo (S'01) received the M.S. degree in computer science (Hons.) from the University of Murcia (UMU), Murcia, Spain, in 1999 and the M.S. and Ph.D. degrees in electrical engineering from Columbia University, New York, in 2003 and 2007, respectively.

Currently, he is with Telefonica Research, Barcelona, Spain. His research interests are in the areas of wireless networking and cross-layer design.

Dr. Toledo received Spain's National Academic Excellence Award, the Edwin Howard Armstrong Memorial Award, and the La Caixa Foundation and Rafael del Pino Foundation fellowships.

Xiaodong Wang (S'98–M'98–SM'04–F'08) received the Ph.D. degree in electrical engineering from Princeton University, Princeton, NJ.

Currently, he is with the faculty of the Department of Electrical Engineering at Columbia University, New York. His research interests include computing, signal processing, communications, wireless communications, statistical signal processing, and genomic signal processing. He has published extensively in these areas. Among his publications is a recent book *Wireless Communication Systems: Advanced Techniques for Signal Reception* (Prentice-Hall, 2003). He has served as an Associate Editor for the IEEE TRANSACTIONS ON COMMUNICATIONS, IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, IEEE TRANSACTIONS ON SIGNAL PROCESSING, and IEEE TRANSACTIONS ON INFORMATION THEORY

Dr. Wang received the 1999 National Science Foundation CAREER Award, and the 2001 IEEE Communications Society and Information Theory Society Joint Paper Award.