Robust Detection of Selfish Misbehavior in Wireless Networks

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

***Abstract*— The CSMA/CA protocols are designed under the assumption that all participant nodes would abide to the protocol rules. This is of particular importance in distributed protocols such as the IEEE 802.11 distributed coordinating function (DCF), in which nodes control their own backoff parameters. In this work, we propose a method to detect selfish misbehaving terminals that may deliberately modify its backoff window to gain unfair access to the network resources. We develop nonparametric batch and sequential detectors based on the Kolmogorov-Smirnov (K-S) statistics that do not require any modification on the existing CSMA/CA protocols, and we apply it to detect misbehaviors in an IEEE 802.11 DCF network using the ns-2 simulator. We compare the performance of the proposed detectors with the optimum detectors with perfect information about the misbehavior strategy, for both the batch case (based on the Neyman-Pearson test), and the sequential case (based on Wald's sequential probability ratio test). We show that the proposed nonparametric detectors have a performance comparable to the optimum detectors for the majority of misbehaviors (the more severe) without any knowledge of the misbehavior strategies.**

***Index Terms*— CSMA/CA, MAC misbehavior, Kolmogorov-Smirnov test, IEEE 802.11.**

I. Introduction

THE CARRIER-SENSE 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 many nodes in a network. This contention resolution is typically based on cooperative protocols (such as the IEEE 802.11 distributed coordination function, DCF), and it is one of the most popular protocols for wireless networks. The operation of the protocol is based on the time multiplexing access of the terminals, and in the trust that all the nodes will obey the protocol guidelines. However, the pervasive nature of wireless networks together with the requirement for flexible and readily reconfigurable protocols has led to the extreme where wireless network devices have become easily programmable [1]. Terminals can easily modify their wireless interface software and change the protocol parameters for their own benefit, to gain advantage over other peers (selfish misbehavior), or to disrupt the network operation (denial-of-service, DoS). It is important, then, to have a mechanism to detect when a terminal is not abiding by the protocol rules. Detecting misbehavior, however, is not an easy task. The main difficulty comes from the random operation of the CSMA/CA protocol, and is exacerbated by the nature of the wireless medium itself, where channel impairment and interference make network conditions to appear different for different terminals.

Deviation from legitimate protocol operation in wireless networks has received considerable attention from the research community in recent years. Most of the current research deals with the case of "malicious" attacks, in which terminals do not obey the protocols with the sole objective of disrupting the operation of the network, even in their own determent. Malicious misbehaviors of this kind are often referred to as denial-of-service attacks. These and other security issues in wireless networks can be found in [2].

While malicious misbehavior is abundant, it lacks a clear incentive from the attacker and hence it is usually limited to a small percentage of users. "Selfish" misbehaviors, on the other hand, are inflicted by users who wish to increase their own share of the common transmission resources; these users are rational, and not malicious [3]. Selfish terminals are often analyzed in the framework of game theory, as they compete to maximize their own utilities [4] [5] [6]. A typical selfish misbehavior may include terminals that refuse to forward packets on behalf of other hosts to conserve energy, or terminals that knowingly modify protocol parameters to gain unfair access to the channel. The threat of a selfish terminal is more credible, as every terminal in the network has a clear incentive to misbehave. The prompt detection of such misbehaving is a major security issue. In fact it is shown in [7] that an IEEE 802.11 DCF can be designed with complete stability (i.e., free of misbehavior) if there exists a way to detect terminals that deviate from the protocol in a prompt way. In this work we present robust nonparametric batch and sequential detector based on the Kolmogorov-Smirnov (K-S) statistics that does not require any modification on the existing CSMA/CA protocols, so it can be used by any terminal in the network that monitors the transmissions. Our method is robust, i.e., it would detect any deviation from the protocol even in the presence of an intelligent attacker. Moreover, it can be adjusted to provide specific false alarm and detection probabilities, being applicable to a wider range of scenarios.

Protocol misbehavior has been studied in various scenarios in different communication layers and under several mathematical frameworks. Most notably a heuristic set of conditions is proposed in [8] for testing the extent to which MAC protocol parameters have been manipulated. The heuristic nature of the method limits its application to specific protocols, and the

Manuscript received July 1, 2006; revised February 15, 2007. 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 N00014-03-1-0039. Alberto Lopez Toledo was supported by Fundacion Rafael del Pino.

The authors are with the Department of Electrical Engineering, Columbia University, 500 West 120th Street, New York, NY, USA (e-mail: {alberto,wangx}@ee.columbia.edu.).

Digital Object Identifier 10.1109/JSAC.2007.070807.

technique can be compromised by any malicious user that knows those conditions. Moreover, it is difficult to determine its statistical performance (e.g. detection or false alarm probabilities), and it relies in expert knowledge in order to set up and maintain the set of heuristic rules. In [3], a modification to the IEEE 802.11 MAC protocol is proposed to detect selfish misbehavior. The approach assumes a trustworthy receiver, since the latter assigns to the sender the back-off value to be used. Relying on the receiver and modifying the IEEE 802.11 protocol represent its major drawbacks. Finally, in [1], a minimax detection framework is employed to analyze the instance of theoretical worst-case attacks. The approach is more robust, but no operational method to detect misbehavior is proposed. It assumes, in the first place, that any transmission by a terminal is observable. However, only the $successful$ transmissions are observable, and in the event of a collision, it is not possible to determine what terminals where involved in it. Second, it assumes an ideal scenario in which the misbehavior strategy is completely known. For those reasons, the detection mechanism proposed in [1] can not be used in practice. Good discussions can be found in [3] and [1] on other techniques to treat misbehavior in ad-hoc networks [9], fairness [10] and other layers other than the MAC layer [11].

The remainder of this paper is organized as follows. Section II describes the CSMA/CA protocol and its vulnerability to misbehavior. In Section III we present the problem formulation and give a formal characterization of misbehavior in CSMA/CA. We propose our misbehavior detection algorithms in Section IV. The performance of the algorithms is evaluated in Section V by using realistic ns-2 simulations. Finally, Section VI concludes the paper.

II. CSMA/CA Protocol and IEEE 802.11 DCF

CSMA is a contention-based MAC protocol in which the transmitting terminal senses the shared medium before transmitting. The rationale behind it is to avoid more than one terminal transmitting at the same time, hence avoiding collisions. If the channel is sensed busy or a collision occurs, the terminal waits for a randomly chosen period of time (backoff), and then checks again to see if the channel is clear. Collision avoidance (CA) is used to improve the performance of CSMA by attempting to reserve the network for a single transmitter. This is important for systems in which collisions are very costly, such as in wireless networks, where they cannot be detected by the terminals producing them.

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: the *basic access* and the *RTS/CTS* access [7].

Basic Access: 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 only allowed to transmit 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 $v \in [CW_{\min}, CW_{\max}]$, where $CW_{\min}$ and $CW_{\max}$ are called the *minimum* and *maximum* contention windows respectively. At the first transmission attempt v is set to $CW_{\min}$. 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 busy, the backoff timer is paused until the channel is sensed idle again. When the backoff timer reaches 0, the terminal is allowed to transmit. 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 $v = CW_{\min}$. This is referred to as a "heavy decrease" in [12]. 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 $CW_{\max} = 2^m CW_{\min}$, where m is usually referred to as the *maximum backoff stage* [12]. This mode of operation is often referred to as a *binary exponential* scheme. The values of $CW_{\min}$, $CW_{\max}$ and the slot size are determined by the characteristics of the physical layer.

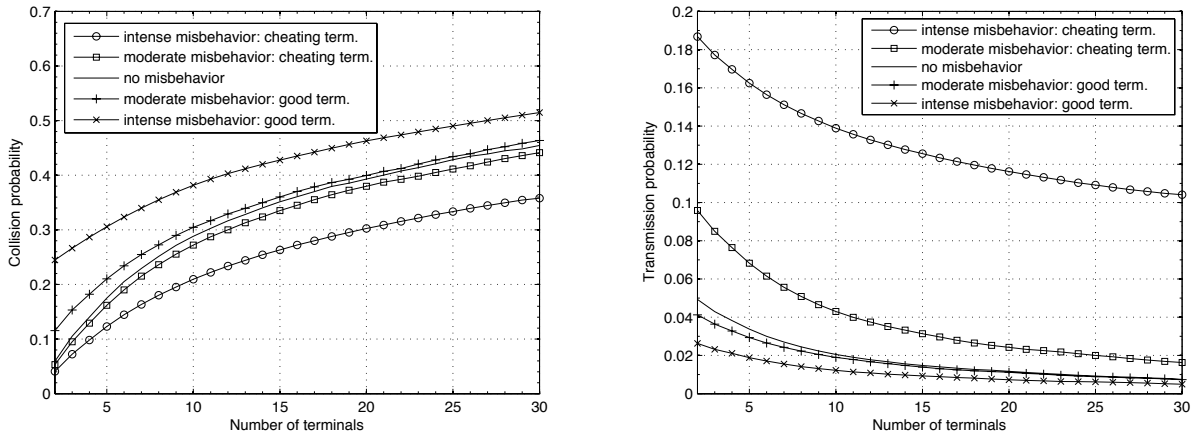
The *RTS/CTS* access is similar to the *basic access*, but it makes use of a four-way handshake protocol in which prior to data transmission a terminal transmits a special short request-to-send frame (RTS) to try to reserve the transmission and reduce the cost of collisions.

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.

B. Effect of Selfish Misbehavior in IEEE 802.11 DCF

Given its distributed nature, the IEEE 802.11 DCF bases its operation on the individual terminals correctly assigning their backoff intervals according to the protocol. In the absence of a central controlling unit enforcing this policy, a selfish terminal might try to select small backoff intervals to gain a significant advantage in channel access probability over time. By increasing their transmission probabilities, selfish terminals produce an increment in the number of collisions in the network, forcing the rest of (well-behaved) terminals to, in turn, increment their backoff intervals, further increasing the advantage for the selfish terminals. Typical techniques of a selfish terminal include reducing their contention window, selecting a smaller minimum window size or using a different scheme instead of the binary exponential (e.g., waiting for a fixed amount of time). For the rest of the paper we consider a terminal to be operating correctly if it uses the binary exponential protocol described in Section II-A, with $CW_{\min} = 32$ and $CW_{\max} = 1024$.

The effect of a misbehaving terminal can be drastic to the operation of the protocol, especially for the well-behaved ter-



(a) Observed collision probabilities.

(b) Observed transmission probabilities.

Fig. 1. Effect of having just one misbehaving node in an IEEE 802.11 DCF network.

minals as the following example illustrates. Figs. 1(a) and 1(b) show the collision probability and transmission probability of an IEEE 802.11 DCF network with one misbehaving node that either uses a fixed backoff window with $CW_{\min} = CW_{\max} = 8$ (an *intense* misbehavior), or uses the binary exponential protocol with $CW_{\min} = 16$ and $CW_{\max} = 1024$ (a *moderate* misbehavior). Each point in the figures was obtained by performing 1000 runs in the ns-2 simulator for different number of terminals saturating in the network, using the parameters and scenario described in Section V-A. As we can see, the misbehaving terminal will observe a much reduced collision probability, resulting in a share of the medium as high as 5 times of those of the well-behaved terminals. The difference is notable even for more moderate misbehaviors, hence there is a strong incentive for a node to misbehave, as the potential benefit in terms of increased throughput is high even for a small deviation from the protocol.

III. Problem Formulation

A. Hypothesis Test

The main and most important characteristic of a selfish misbehavior is that the strategy used by the misbehaving terminal is completely unpredictable. This uncertainty makes the problem of detecting misbehaving terminals a difficult one. Our objective is to develop a method to detect when a terminal is misbehaving by observing its operation in the network, and we do it following the signal detection framework [13].

Let $x_1, ..., x_K$ be a sequence of observations related to the operation of a CSMA/CA terminal. We consider two hypotheses, the *null* hypothesis H_0 corresponds to the observed terminal *not misbehaving*, while the alternate hypothesis H_1 corresponds to the case that the terminal *is misbehaving*. We bias towards the not misbehaving case because the cost of a false alarm is high, as it is more important to guarantee that the well-behaved terminals are not accused of misbehaving (and potentially being disconnected from the network). We write this problem as

$$\text{choose} \begin{cases} H_0 : x_1, ..., x_K \sim f_0 \\ H_1 : x_1, ..., x_K \sim f_1, \end{cases} \quad (1)$$

where f_0 and f_1 are the probability distributions of the observations when a node is not misbehaving and misbehaving respectively. We want to design a decision rule $\delta(x_1, ..., x_K) \in \{0, 1\}$ to discriminate between the two hypotheses.

With the above formulation two questions arise. First, what are the possible observations $x_1, ..., x_K$, and more importantly, if there exists a way to take advantage of those observations to identify misbehavior? Second, what are the probability distributions of the observations for the non-misbehaving case f_0, and for the misbehaving case f_1? We refer to these distributions as the *strategy* of a terminal.

The detection of a misbehaving terminal faces several problems. The random operation of the CSMA/CA protocol makes it difficult to distinguish a well-behaved terminal that has randomly selected small backoff intervals from a misbehaving terminal that has *deliberately* selected small backoffs. This is exacerbated by the inherent volatile nature of the wireless medium itself [1]. If misbehavior is detected, the observer terminals should have a mechanism to inform the rest of the network, e.g., the access point, in order to take appropriate actions. The specifics of such a mechanism, however, fall outside the scope of this paper.

B. The Measurements

A common approach in the study of misbehavior and unfairness makes use of the difference in throughput observed by different terminals. The problem with this approach is that throughput is not a part of the protocol itself, but just a consequence (e.g., CSMA/CA does not define that terminals should have certain throughput, or even that all of them should share the medium fairly, although it is certainly expected). Moreover, channel conditions may cause certain terminals to have more throughput than others while still abiding by the protocol. Also, throughput measurements depend on system parameters such as slot size and access mode that are in turn dependent on the specific characteristics of the underlying protocol.

The IEEE 802.11 DCF does not specify throughput, but only defines the method to select the backoff intervals. We

$$f_0(x_i) = \begin{cases} \mathcal{U}[0,32] & \text{w.p.} \quad 1-p_c \\ \mathcal{U}[0,32]+\mathcal{U}[0,64] & \text{w.p.} \quad p_c(1-p_c) \\ \mathcal{U}[0,32]+\mathcal{U}[0,64]+\mathcal{U}[0,128] & \text{w.p.} \quad p_c^2(1-p_c) \\ \mathcal{U}[0,32]+\mathcal{U}[0,64]+\mathcal{U}[0,128]+\mathcal{U}[0,256] & \text{w.p.} \quad p_c^3(1-p_c) \\ \mathcal{U}[0,32]+\mathcal{U}[0,64]+\mathcal{U}[0,128]+\mathcal{U}[0,256]+\mathcal{U}[0,512] & \text{w.p.} \quad p_c^4(1-p_c) \\ \mathcal{U}[0,32]+\mathcal{U}[0,64]+\mathcal{U}[0,128]+\mathcal{U}[0,256]+\mathcal{U}[0,512]+\sum_{i=5}^{n_c}\mathcal{U}[0,1024] & \text{w.p.} \quad p_c^{n_c}(1-p_c), \end{cases} \tag{4}$$

are interested in knowing the sequence of backoff intervals selected by a given terminal, in particular, how many idle slots the terminal waited since its last transmission before attempting a new transmission, so that we can then check if that sequence corresponds to the case of a binary exponential increase with the correct $CW_{\min}$ and $CW_{\max}$ parameters. However, the sequence of backoff intervals selected by a terminal, i.e., its transmission *attempts*, is not directly observable in a CSMA/CA system, and in particular in an IEEE 802.11 system, because the only observable transmissions from a terminal are *successful* transmissions. Attempted transmissions that result in collisions can be observed, but it is not possible to distinguish which terminals *are involved* in them. So our observation events are the specific times at which a given terminal transmits successfully.

Because the terminals only decrement their backoff counters when the channel is idle, we focus on the number of idle slots between two consecutive successful transmissions of a certain terminal. We can use the procedure in [1] to obtain the number of idle slots in the network. Consider the RTS/CTS access in the IEEE 802.11 DCF. Let t_{i-1} be the end time of the last transmission of any terminal and let t_i be the time of the current RTS packet reception. We assume that those events are observable from all the nodes in range. The number of idle slots between those events can be calculated as

$$x_i = \frac{t_i - t_{i-1} - T_{DIFS} - T_O}{\sigma}, \quad i > 1, \tag{2}$$

where T_{DIFS} is the duration of the $DIFS$ frame, σ is the duration of an idle slot, and T_O is the duration of transmissions from other terminals and collisions, including their interframe times. A terminal that is not directly in range of the terminal that transmits the RTS frame can also compute the backoff by using as a reference the time of reception of the overheard ACK from the receiver of the previous data segment as follows:

$$x_i = \frac{t_i' - T_{ACK,i-1} - T_{DIFS} - T_{RTS} - T_{SIFS} - T_O}{\sigma}, \quad i > 1. \tag{3}$$

where t_i' is the time of reception of the CTS packet, $T_{ACK,i-1}$ is the duration of the previous ACK frame and T_{DIFS}, T_{SIFS} and T_{RTS} are the durations of a DIFS and SIFS periods, and the RTS frame respectively, and T_O is defined as in (2). The number of idle slots for the case of basic access can be obtained similarly.

C. Probability Distribution of Legitimate Terminals

To calculate the distribution of the samples x_i under H_0, we consider a typical IEEE 802.11 DCF, where $CW_{\min} = 32$ and $CW_{\max} = 1024$. We derive the distribution f_0 of the number of idle slots a terminal would wait between successful transmissions as follows. Although the possible back-off values are discrete, for simplicity we use continuous distributions to facilitate mathematical treatment. Let us assume that the legitimate terminal is saturating, i.e., it always has a packet to send, and let p_c be the probability that the terminal will suffer from a collision if it transmits in the current slot. After a successful transmission of a terminal, the next attempt to transmit will happen after τ_1 idle slots where $\tau_1 \sim \mathcal{U}[0,32]$ and $\mathcal{U}$ denotes the uniform probability distribution. That transmission will be successful with probability $(1-p_c)$, and hence $x_i = \tau_1$. If there is a collision, with probability p_c, then the terminal would double its window size and make another attempt after $\tau_2 \sim \mathcal{U}[0,64]$ slots. If that last transmission is successful then the number of idle slots after the last successful transmission is $x_i = \tau_1 + \tau_2 \sim \mathcal{U}[0,32] + \mathcal{U}[0,64]$ with probability $p_c(1-p_c)$. Following the above argument we can easily obtain the distribution of the number of idle slots between successful transmissions, $f_0(x_i)$, assuming p_c does not vary between successful transmissions, as given in (4), where $n_c < I_{\max}$ is the number of collisions that the transmission undergoes and $I_{\max}$ is the maximum allowable number of collisions. Fig. 2 shows the probability density function given in (4), and the histogram of the number of idle slots between successful transmissions in an IEEE 802.11 DCF network with 10 saturating terminals obtained using the ns-2 simulator with the parameters and scenario described in Section V-A. Note that f_0 has the support $[0, \sum_{i=0}^{m} 2^i CW_{\min} + \sum_{i=I_{\max}-m}^{I_{\max}} 2^m CW_{\min}]$, where m is maximum backoff stage. As the figure shows, the analytical model presented in (4) matches perfectly with the empirical data obtained from the simulations.

We denote the pdf f_0 calculated above as the strategy of a saturating legitimate terminal, that is, a terminal that follows the protocol and always has a packet to send. The calculation of f_0 requires the estimation of the collision probability p_c for the legitimate terminals, which will be discussed in Section IV-A.

D. Characterizing Misbehaving Terminals

Unlike the strategy of a legitimate terminal, the unknown strategy of a (potentially) misbehaving terminal is not unique. Let us define f_1 as the unknown strategy of the observed terminal for which we are interested in determine whether or not it is misbehaving. In order to characterize and quantify misbehavior we will compare f_1 to the strategy of a saturating legitimate node f_0. Denote $F_1(x)$ and $F_0(x)$ as the cumulative distribution functions (cdf) for f_1 and f_0 respectively.

Let us focus exclusively on the terminals that abide by the protocol rules. Those terminals are obviously not misbehaving. If the terminal is saturating, then it is clear that $F_1(x) =$

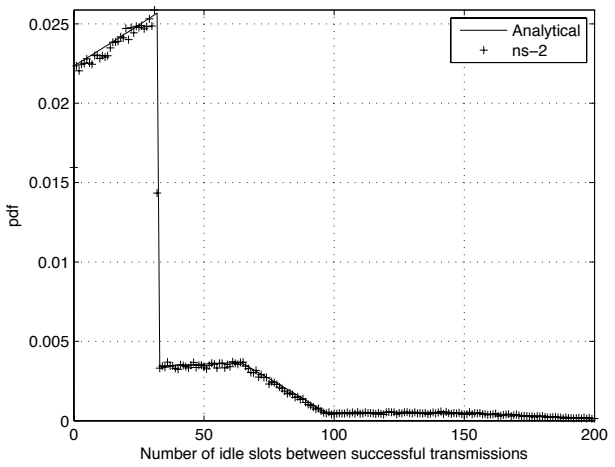


Fig. 2. The pdf of the number of idle slots between successful transmissions for an IEEE 802.11 DCF network with 10 nodes.

$F_0(x)$. If the terminal is not saturating, e.g., it rests for a unknown time $\Delta > 0$ after each or some transmissions, then obviously its cdf satisfies $F_1(x) < F_0(x)$, $\forall x$. In general, for any terminal using the correct protocol, either saturating or not, we have $\forall x$, $F_1(x) \leq F_0(x)$. Intuitively, if the cdf of a terminal is always on or below the cdf of a well behaved terminal that is always transmitting, then the terminal is definitely not misbehaving [14].

The above discussion leads to our definition of misbehavior: a terminal using a unknown strategy f_1 with cdf F_1 is misbehaving, if $\exists x$, s.t. $F_1(x) > F_0(x)$, where F_0 is the cdf of the strategy of a legitimate terminal that is saturating.

Note that our definition of misbehavior does not take into account the transmission probability (and hence, the throughput) of the terminals. It is easy to find a terminal satisfying $\exists a\ F_1(a) > F_0(a)$ and $\forall x \neq a\ F_1(x) \leq F_0(x)$, that has a transmission probability lower than that of the legitimate saturating terminal, and therefore appears non-misbehaving. However, the CSMA/CA protocol is designed so that the transmissions of a terminal are distributed as uniformly as possible in time to avoid collisions. Fairness is achieved as long as every terminal uses the same strategy. A terminal transmitting less than a legitimate terminal but using a different strategy may produce a disruption in the service at its transmission attempts, perturbing the normal operation of the protocol. Those terminals should be considered as misbehaving terminals. Our definition of misbehavior is general enough to capture this often overlooked type of misbehavior.

IV. Misbehavior Detection Strategies

We are interested in developing a detector that can discriminate between a legitimate terminal using f_0 and a misbehaving terminal that does not.

It is well known that the optimal decision rule for the binary hypothesis test problem in (1) involves computing the likelihood ratios

$$S_K = \log \frac{f_1(x_1, ..., x_K)}{f_0(x_1, ..., x_K)}, \tag{5}$$

and comparing it with a threshold. Unfortunately, the likelihood ratio test cannot be used to detect the presence of *any* arbitrary misbehavior because the distribution f_1 of the number of idle slots between successful transmissions of a potential misbehavior, or any other parameter about its operation is unknown. It is necessary then to use *distribution-free* or *nonparametric* approaches to perform the detection. In what follows we will present a nonparametric test based on the Kolmogorov-Smirnov statistic, that fits seamlessly with our definition of misbehavior using a fixed number of samples, and then we will propose a sequential version of the test.

A. Collision Probability Estimation

As seen in Section III, in order to obtain the distribution f_0 of the idle slots between successful transmissions for a saturating legitimate terminal, the probability of collision in the network p_c has to be estimated. A terminal can keep track of its own transmissions and count how many of them resulted in collisions. That is, an estimate of the collision probability is given by

$$\hat{p}_c = \frac{C}{T} \tag{6}$$

where T is the number of transmission attempts and C is the number of those resulting in collisions. The terminal may increase or decrease the observation interval T to estimate the varying collision probability p_c more accurately.

The above approach requires the measuring terminal to transmit in order to count the number of transmission attempts that result in collision, which makes it not suitable for terminals that do not have anything to send. Moreover, if a misbehaving terminal is present in the network, the transmission rate of a legitimate terminal could be much lower than that of the misbehaving terminal, so the above estimator could be too slow compared to the transmission rate of a misbehaving terminal. A faster estimate can be obtained if a terminal does not count how many of its own transmissions resulted in collisions, but instead how many of the total number of transmissions in the network resulted in collisions. While it is not possible to observe how many terminals attempted a transmission for any give collision, because the identity and the number of the colliding terminals is hindered by the collision itself, the average number of terminals colliding (*collision factor*) γ is a function of the protocol and the number of terminals competing in the network. For example, Fig. 3 shows the average number of terminals involved in a collision in a standard IEEE 802.11 DCF network with $CW_{\min} = 32$ and $CW_{\max} = 1024$ when all the terminals are well behaved. The figure was obtained by performing 1000 runs in the ns-2 simulator using the simulation parameters described in Section V-A for each value of U. We define this new estimator as

$$\tilde{p}_c = \frac{C\gamma}{T' + C\gamma}, \tag{7}$$

where C is the number of collisions, T' is the number of $successful$ transmissions observed by the terminal in the network, and γ is the collision factor. Note that $T' + C\gamma$ is the average number of transmission attempts in the network, and $C\gamma$ is, on average, how many of them resulted in collisions.

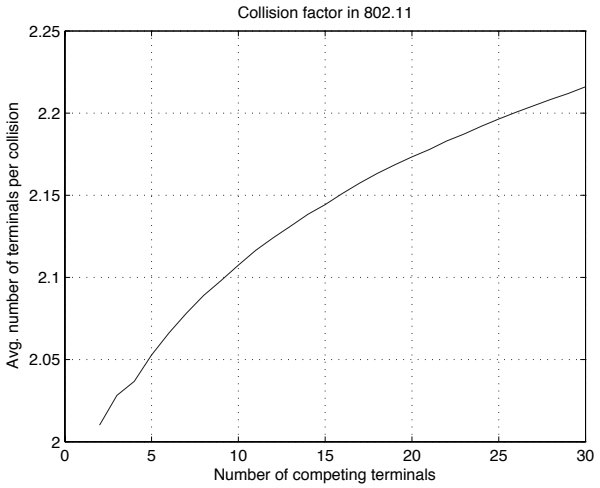


Fig. 3. Collision factor of IEEE 802.11 DCF.

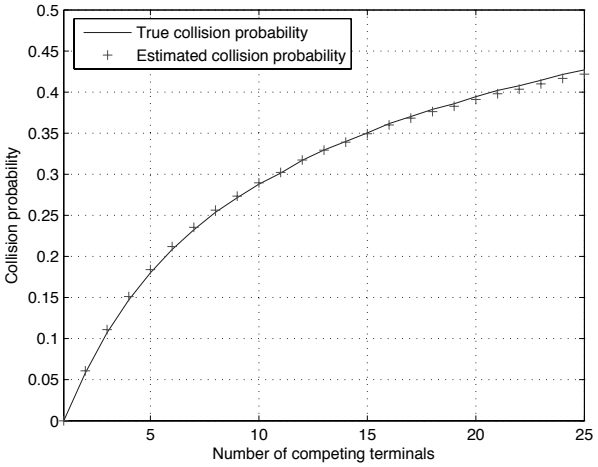


Fig. 4. Estimated collision probability using the collision factor for $\gamma = 2.14$.

As before, the measuring terminal may increase or decrease the observation interval T'. In our simulations we set $T' = 30$.

The estimate $\tilde{p}_c$ requires the use of the appropriate γ corresponding to the number of competing terminals U. While U can be estimated with techniques such as those in [15], in an IEEE 802.11 DCF network with $U \leq 25$[1], we have observed that selecting a fixed $\gamma = 2.14$ results in a very good approximation $\tilde{p}_c \approx p_c$ for any given U. Fig. 4 shows the true collision probability and the estimated collision probability using (7) with $\gamma = 2.14$, obtained in the ns-2 simulator using the parameters described in Section V-A. The small error in the estimation of the collision probability has virtually no effect on the false alarm probability of the detectors, and simplifies the system, as U does not have to be estimated. This approximation, however, comes at the cost of reducing the probability of detection, although only for misbehavior cases that are extremely close to the legitimate operation of the protocol, which are undoubtedly of less interest.

It is expected that the true collision probability p_c varies slowly with respect to the time scale of the transmissions by any terminal in the network. However, the noise in the estimates $\hat{p}_c$ and $\tilde{p}_c$ may overshoot p_c so that a legitimate terminal may appear as misbehaving. This is of no concern when there is a misbehaving terminal in the network, as the difference in collision probability is larger than the noise, however it affects the false alarm probability. Because the cost of a false alarm is very high, we filter the data to reduce the noise, using a robust locally weighted polynomial regression model (rloess)[2] with smoothing parameter of 5.

Finally, let $\tilde{p}_c^{(1)}, .., \tilde{p}_c^{(q)}$ be the sequence of collision probabilities estimated using (7) and filtered as indicated above. The cdf of the number of idle slots between successful transmissions for a legitimate saturating terminal in that period can be calculated as the average of the cdfs for each of the observed $\hat{p}_c^{(i)}$, i.e.,

$$\hat{F}_0 = \frac{1}{q} \sum_{i=1}^{q} F_0 \left(\tilde{p}_c^{(i)} \right), \tag{8}$$

where $F_0(p_c^{(i)})$ is the cdf of $f_0(p_c^{(i)})$, which is calculated using (4) with collision probability $p_c^{(i)}$.

B. The Kolmogorov-Smirnov Test

The Kolmogorov-Smirnov (K-S) test [16], [17], is the most widely used goodness-of-fit test for continuous data. It is based on the empirical distribution function (edf), which converges uniformly almost surely to the real population cdf (Glivenko-Cantelli Theorem) [18]. The K-S test determines whether the underlying distribution f_1, from which samples are drawn, differs from an hypothesized distribution f_0. The K-S test compares the edf $\hat{F}_1$ obtained from the data samples with the hypothesized cdf F_0, and determines whether $F_1 = F_0$, or $F_1 < F_0$, or $F_1 > F_0$. For the misbehavior detection problem, we define the null hypothesis as the event where a node is not misbehaving, and hence we will use the one-sided test

$$\text{choose} \begin{cases} H_0 : F_1 \leq F_0 & \text{(not misbehaving)} \\ H_1 : F_1 > F_0 & \text{(misbehaving).} \end{cases} \tag{9}$$

Let $x_1, x_2, ..., x_K$ be the observations of the number of idle slots between successful transmissions from a terminal using an unknown strategy f_1. The edf of the observations is given by

$$\hat{F}_1(x) = \frac{1}{K} \sum_{i=1}^{K} \mathbb{1} \{x_i \leq x\}, \tag{10}$$

where $\mathbb{1}(\cdot)$ is the indicator function.

The one-sided K-S test statistic D is defined as the maximum value of the difference between the two cdfs as

$$D \triangleq \max_{-\infty < x < +\infty} \{F_1(x) - F_0(x)\}, \tag{11}$$

and can be calculated as

$$\hat{D} = \max_{1 \leq i \leq K} \left\{ \hat{F}_1(x_i) - \hat{F}_0(x_i) \right\}, \tag{12}$$

[1]Note that U is the number of terminals that are $simultaneously$ sending at any given point in time, not the total number of terminals in the network. So $U \leq 25$ is a reasonable assumption.

[2]We used the implementation provided by MATLAB, because it is widely available; and the function `smooth` with parameter `rloess`, as it provided the best empirical results in our tests.

Algorithm 1 K-S test for a fixed number of samples with $P_{FA} = \alpha$

1: Calculate K observations of the number of idle slots between transmissions of the observed terminal $x_1, ..., x_K$, using (2) or (3). Calculate the edf $\hat{F}_1$ from the samples of the observed terminal using (10).
2: Simultaneously, collect the estimates $\tilde{p}_c^{(1)}, .., \tilde{p}_c^{(q)}$ using (7), and filter as indicated in Section IV-A. Calculate the cdf of a legitimate terminal $\hat{F}_0$ using (8).
3: Perform the one sided K-S test for $\hat{F}_1 > \hat{F}_0$ and obtain the significance level P using (14).
4: **if** $P \leq \alpha$ **then**
5: reject H_0. The terminal is misbehaving.
6: **else**
7: do not reject H_0. The terminal is not misbehaving.
8: **end if**

where $\hat{F}_0$ and $\hat{F}_1$ are given respectively by (8) and (10).

Define [16]

$$\lambda(\hat{D}) = \max\left\{\left(\sqrt{K} + 0.12 + \frac{0.11}{\sqrt{K}}\right)\hat{D}, 0\right\}. \quad (13)$$

Then, the hypothesis H_0 is rejected at a significance level α if $P(D > \hat{D}) < \alpha$, where [19]

$$P(D > \hat{D}) = e^{-2\lambda(\hat{D})^2}. \quad (14)$$

It is known that the Kolmogorov-Smirnov statistic is more sensitive near the center of the distribution than at the tails. Several methods have been proposed that solve this limitation, such as the Anderson-Darling test [19]. However, the distribution of the Anderson-Darling statistic is not known for an any arbitrary distribution. For this reason there is not a similar expression for its significance such as (13)-(14), and so we do not use it in this work. It is also shown in [18] that the power of the K-S test can be improved by a slight modification in the calculation of the edf, by applying Bloom's 'α,β-correction'. The two-stage δ-corrected K-S test uses these modifications. We have investigated the performance of the modified test, and have observed that the performance gain is negligible, if at all, in our particular scenarios where $K > 20$ and the probability of detection $P_D > 0.9$. For these reasons we will simply employ the standard K-S test.

We can now give the algorithm to test if a terminal is misbehaving: fixing the number of samples K, the measuring terminal calculates a new x_i using (2), (3) for each successful transmission of the observed terminal. Simultaneously, it calculates a new estimate of the collision probability $\tilde{p}_c^{(j)}$ using (7), every T' successful transmissions in the network (from *any* terminal). After the K-th successful transmission of the observed terminal, the algorithm uses the collected sequence of idle slots between successful transmissions $x_1, ..., x_K$ and the calculated sequence of estimates of the probability of collision $\tilde{p}_c^{(1)}, .., \tilde{p}_c^{(q)}$, to perform the hypothesis test in (9) with a false alarm probability $P_{FA} = \alpha$. Algorithm 1 describes the K-S test in detail.

The algorithm can be used by any terminal in the network. A typical scenario would involve the IEEE 802.11 DCF access point (AP) collecting samples x_i and implementing the algorithm for each terminal. Upon the detection of a misbehaving terminal, the AP can take appropriate actions, such as disconnecting the offended terminal from the network for a period of time. In the case of an ad-hoc network, terminals may implement the algorithm on their neighbors and, for example, deny forwarding privileges to those misbehaving. Our detector can also be used to implement the Nash equilibrium described in [7], where it is shown that it is possible to dissuade terminals from misbehaving by using a punishing strategy as long as there exists a reliable detection mechanism to detect misbehavior. Our mechanism is not affected by the presence of intelligent attackers that collude, as the detection can be performed by any terminal in the network, even those not involved in transmissions. The colluding attackers would be detected independently by any terminal monitoring their transmissions. Finally, network conditions such as interference do not negatively affect the detection, as the measuring terminal takes the network conditions into account by continually estimating the probability of collision p_c in the network. Any terminal should use the legitimate strategy corresponding to the probability of collision observed in the network, wether that collisions are produced by other terminals transmitting, or are caused by external interference.

C. N-truncated Sequential K-S test

Given the the sequential nature of the problem, in which a terminal acquires observations from other terminals in a sequential manner, it is also of interest to approach the misbehavior detection problem sequentially. That is, instead of using a fixed number of samples for each decision, some realizations of the observation sequences may allow us to make decision after only few samples, whereas for other realizations we may need to continue sampling to make a decision.

While the algorithm described in the previous section is designed for a fixed number of samples, the significance values of the K-S statistic (14) allows us to make sequential decisions on the null hypothesis. With each new sample, the edf $\hat{F}_1$ can be updated and the K-S statistic reevaluated.

A sequential K-S test can be formed by concatenating N K-S tests, starting with 1 sample and adding one sample at each subsequent stage up to N stages, where N is the truncation point of the test. We fix the desired false alarm probability of the sequential test to $P_{FA} = \alpha$. Because the sequential test is composed of N tests, we need to calculate the false alarm probability of each stage in order to meet the overall P_{FA}. Let β be the false alarm probability of each stage. Then the resulting P_{FA} of the N-truncated sequential K-S test is

$$\begin{aligned} P_{FA}^{seq} &= \beta + (1-\beta)\beta + (1-\beta)^2\beta + ... + (1-\beta)^{N-1}\beta \\ &= 1 - (1-\beta)^N = \alpha. \end{aligned} \quad (15)$$

So in order to obtain $P_{FA} = \alpha$ in the sequential K-S test, each individual K-S test should use the threshold β where

$$\beta = 1 - \sqrt[N]{1-\alpha}. \quad (16)$$

The sequential algorithm proceeds similarly to Algorithm 1. At stage $n \leq N$ the measuring terminal has already calculated the sequence of idle slots between successful transmissions

Algorithm 2 N-truncated sequential K-S test with $P_{FA} = \alpha$

1: $n = 0$.
2: $\beta \leftarrow 1 - \sqrt[N]{1-\alpha}$.
3: $n \leftarrow n + 1$.
4: Take a new observation x_n of the number of idle slots between transmissions of the observed terminal. Update the edf $\hat{F}_1$ with the new sample x_n using (10).
5: Simultaneously, calculate the new estimates $\tilde{p}_c^{(r+1)}, p_c^{(r+2)}, ..., \tilde{p}_c^{(q)}$ since the last observation x_{n-1} using (7), where $p_c^{(r)}$ is the last estimate from stage $n-1$. Filter the data as described in Section IV-A. Update the cdf of a legitimate terminal $\hat{F}_0$ with those new estimates using (8).
6: Perform the one sided K-S test for $\hat{F}_1 > \hat{F}_0$ and obtain the significance level P using (14).
7: **if** $P \leq \beta$ **then**
8: reject H_0. The terminal is misbehaving.
9: **else if** $n = N$ **then**
10: do not reject H_0. The terminal is not misbehaving.
11: **else**
12: go to 2
13: **end if**

of the observed terminal $x_1, ..., x_{n-1}$, and the sequence of estimates of the collision probability $\tilde{p}_c^{(1)}, .., \tilde{p}_c^{(r)}$. While waiting for the next transmission of the observed terminal, the measuring node calculates new estimates of the collision probability $\tilde{p}_c^{(r+1)}, \tilde{p}_c^{(r+2)}, ..., p_c^{(q)}$, using (7), every T' successful transmissions in the network (from *any* terminal). When the next successful transmission of the observed terminal occurs, the measuring terminal calculates the number of idle slots x_n since the last transmissions, and uses the collected sequences since stage 1, $x_1, ..., x_n$ and $\tilde{p}_c^{(1)}, .., \tilde{p}_c^{(q)}$, to perform the hypothesis test in (9) with a false alarm probability $P_{FA} = \beta$. If a decision is made the algorithm stops, if not it continues with a new sample x_{n+1} until a decision is made or N is reached. In order to avoid having too few estimates of the collision probability (e.g., when n is small), we keep a minimum of 10 estimates obtained before the beginning of the detection that are discarded as new estimates are collected. Algorithm 2 describes the N-truncated K-S test in detail.

Note that the edf calculation of Step 6 can be easily done sequentially, as it only needs to update the edf with a new sample.

As the threshold β used for each stage can be considerably lower than the α used for the batch K-S test, one could think that the detection probability of the sequential K-S test is lower than the batch K-S test. However, as it will be shown in Section V, this is not the case, and the performance of the sequential test is indeed very close to the fixed sample size K-S test.

V. Simulation Results

A. Simulation Setup and Performance Benchmarks

We consider the IEEE 802.11 DCF described in Section II-A, where a legitimate terminal uses $CW_{\min} = 32$ and $CW_{\max} = 1024$. We used an ad-hoc based IEEE 802.11 DCF simulator written in MATLAB to obtain and verify the analytical cdf of the saturating legitimate terminals (4). For the rest of simulations, as well as the figures in the previous sections, we take the observations using the ns-2 network simulator version 2.28 [20], and we implement the detection algorithms using MATLAB. We modified the 802.11 implementation so the nodes measure the observation slots as described in Section III-B for estimating the collision probabilities. The simulated scenario is a 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 the other terminals and runs the detection algorithms over those samples. The parameters used in the simulation are typical for a 11 Mbps 802.11b 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 bytes. The MAC and PHY headers use respectively 272 and 192 bits. The ACK length is 112 bits. 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.

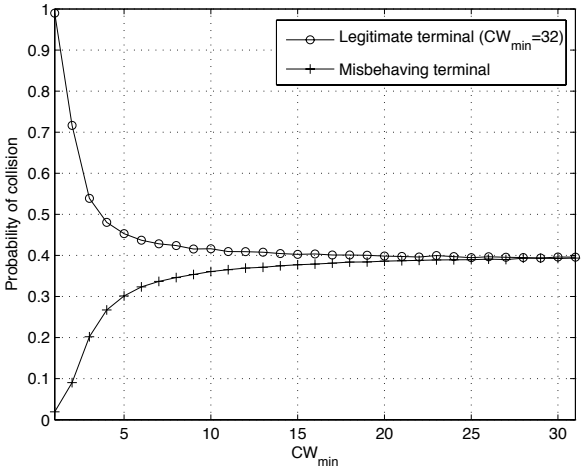


Fig. 5. Probability of collision observed when a misbehaving terminal uses IEEE 802.11 DCF with different $CW_{\min}$.

For simplicity, in our simulations the misbehaving terminals are assumed to use the binary exponential strategy with $CW_{\max} = 2^5 CW_{\min}$, and $CW_{\min} \in \{1, 2, ..., 32\}$. The case of $CW_{\min} = 32$ corresponds to the legitimate terminal. The case of $CW_{\min} = 16$ corresponds to the moderate misbehavior described in Section II-B. Finally the case of $CW_{\min} = 1$ corresponds to a case of extreme misbehavior. Fig. 5 shows the difference in the observed probability of collision in a 20 terminal network when one of them is using the above misbehavior strategies. The figure shows that, as expected, the collision probability observed by the misbehaving terminal is much lower compared to the observed probability of collision of one of the 19 legitimate terminals. Fig. 6 shows the cdf for some misbehavior strategies compared to the strategy of a legitimate terminal when $p_c = 0.1$. All these cases represent

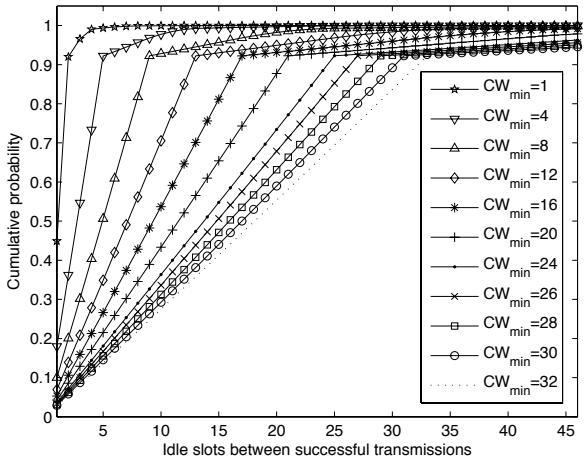


Fig. 6. Cdf of the number of idle slots between successful transmissions when a misbehaving terminal uses IEEE 802.11 DCF with different $CW_{\min}$.

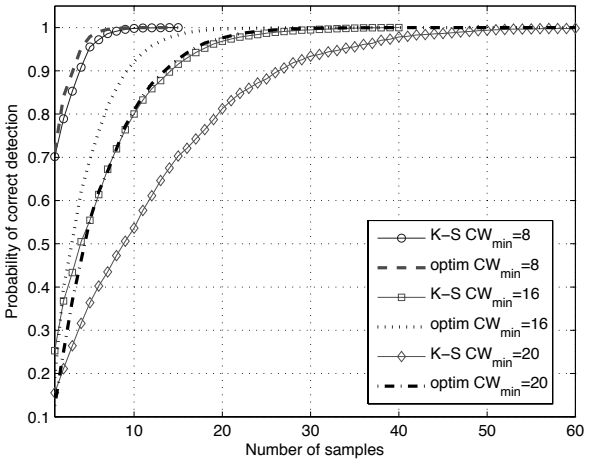


Fig. 7. Performance of the K-S detector vs. optimum detector when a misbehaving terminal uses IEEE 802.11 DCF with different $CW_{\min}$.

a good overview of the different intensity of misbehaviors, and provide a benchmark for the subjective performance of our algorithms (i.e., delay until making a decision). Note that for any misbehavior strategy with $CW_{\min} > 25$ the effect of misbehavior is minimal, so we are interested in a fast detection of the strategies with $CW_{\min} \leq 25$.

In order to test the performance of our algorithms for similar detection and false alarm probabilities, we compare our method to the optimal detectors for both the batch and the sequential case, which correspond to the Neyman-Pearson and the Wald's sequential probability ratio test (SPRT) respectively [13]. While in practice the misbehavior strategy f_1 is not known, we can arbitrarily specify it in our simulations. The performance of the optimal detectors with known f_1 can then serve as the upper bound for the performance of the proposed K-S detectors.

For the batch detector we compare our K-S test with the optimal Neyman-Pearson detector for the same $P_{FA} = \alpha$. For the sequential detector, we compare our N-truncated sequential K-S test with the optimal SPRT detector with the same $P_{FA} = \alpha$ and detection probability P_D.

B. K-S Performance for Fixed Number of Competing Terminals

Consider a network of 10 terminals. Fig. 7 show the probability of detection of our K-S detector, and the optimal Neyman-Pearson detector with perfect information for the misbehavior cases $CW_{\min} = 8$, $CW_{\min} = 16$ and $CW_{\min} = 20$, with $P_{FA} = 0.05$. The K-S detector is able to detect the misbehavior terminals very fast, requiring less than twice the samples needed by the optimum detector with perfect information.

Fig. 8 shows the number of samples needed to detect a misbehaving terminal for different $CW_{\min}$ strategies with $P_{FA} = 0.05$ and $P_D = 0.95$. Note that the performance of the K-S detector starts to degrade only for $CW_{\min} > 29$, which is very close to the strategy of a legitimate terminal.

While the number of samples required grows exponentially as the difference in collision probability is reduced, we are more interested in the class of misbehavior that results in larger gains for the misbehaving terminal. Such a misbehavior would have the most devastating effects on the network, in the sense that it would deny channel access to the other terminals and would lead to unfair sharing of the channel [1]. Hence, it is more valuable, as our detector does, to perform faster for the more severe misbehaviors.

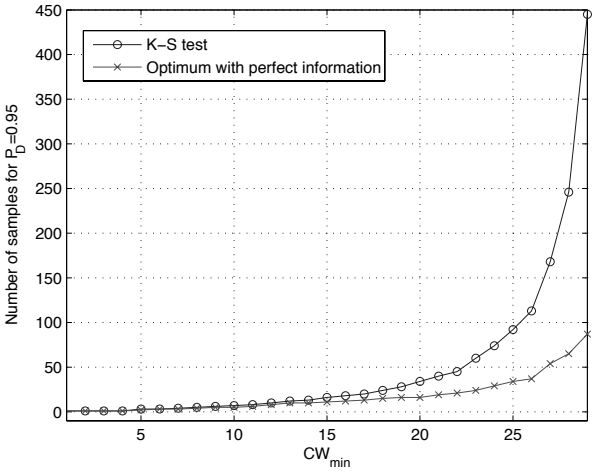


Fig. 8. Number of samples needed to detect a misbehaving terminal using IEEE 802.11 DCF with different $CW_{\min}$ and with $P_D = 0.95$.

Overall, the detection speed of the our K-S detector is high. Under good SNR conditions, a typical IEEE 802.11g network can deliver approximately 24Mbps to the upper layers [21], resulting in an approximate throughput of 2230 packets per second, assuming packets of 1400 bytes. On such a network, and taking into account the throughput of the misbehaving terminal for 10 competing terminals, our K-S algorithm is able to detect the $CW_{\min} = 29$ strategy in slightly less than 2 seconds, and all the misbehavior strategies $CW_{\min} < 29$ in less than a second. These times are comparable to the time a terminal needs to subscribe (and acquire an IP address) to an IEEE 802.11 network.

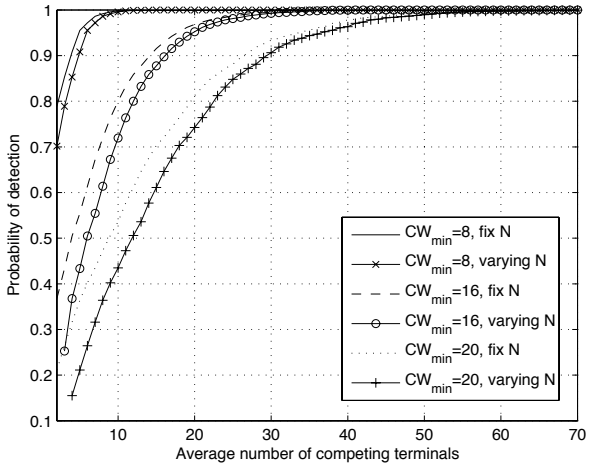


Fig. 9. Performance of the K-S detector for varying number of competing terminals.

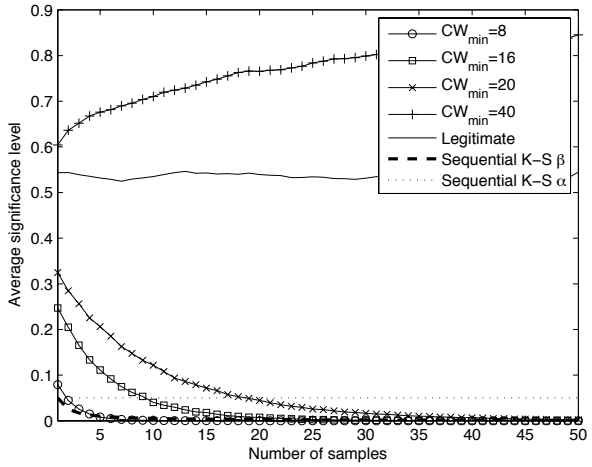


Fig. 10. Significance level of the K-S test as the number of samples increases.

C. K-S Performance for Varying Number of Competing Terminals

Our proposed algorithm can be applied without modification, to a scenario in which the number of competing terminals is changing. When the number of competing terminals changes, the observed probability of collision changes accordingly. However, our algorithm already estimates the probability of collision with an observation window. The approximation is valid under the assumption that the probability of collision does not change between two successive transmissions for a given terminal. As discussed before, the speed of the estimation of p_c is fast enough to make at least one estimation of p_c for each successful transmissions of the observed terminal.

Fig. 9 shows the performance of the K-S detector for an IEEE 802.11 DCF network with one misbehaving terminal using strategies $CW_{\min} = 8$, $CW_{\min} = 16$ and $CW_{\min} = 20$, when the number of competing terminals is fixed, and when the all the terminals (except the one misbehaving) arrive and leave the network with an exponential distribution of parameter 1 second. Note that under such a high rate of change of the number of competing terminals with respect to a typical IEEE 802.11 DCF network, the performance of the detector is very close to the case of a fixed number of terminals for $P_D > 0.95$.

D. Sequential K-S Performance

While the batch test allows us to characterize the performance of the test in the average case, the nature of the problem makes sequential detection more convenient. We use the sequential detector proposed in Section IV-C and we compare it to an optimal SPRT with identical P_{FA} and P_D. While the sequential K-S detector can only fix P_{FA}, as the number of samples tends to infinity, the edf approaches the population hypothetical cdf almost surely, so the power of the test can be increased with the number of samples with the consequent delay cost. However, it is not possible to specify an arbitrary P_D for the test. This is a consequence of the nonparametric nature of the K-S test.

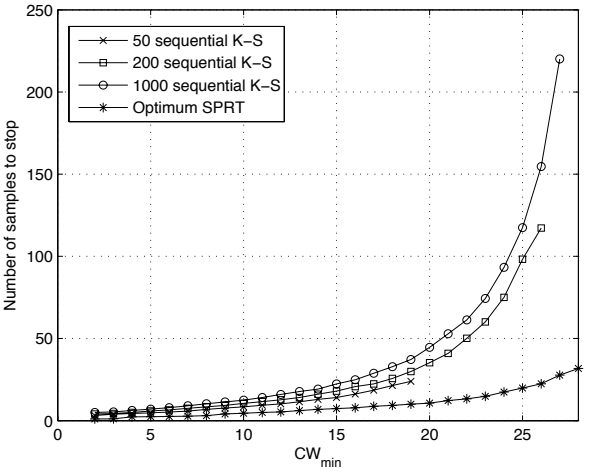


Fig. 11. Stop times of sequential algorithms for $P_D \geq 0.95$.

Fig. 10 shows the significance level of the K-S test for the misbehavior patterns $CW_{\min} = 8$ and $CW_{\min} = 16$, the legitimate strategy $CW_{\min} = 32$ and the non-misbehaving strategy $CW_{\min} = 40$. Note that the significance level converges to 0 when $F_1(x) > F_0(x)$ and it converges to 1 when $F_1(x) < F_0(x)$. Note also that the threshold β of each stage is much lower than α, so it requires more samples for the detection than a batch K-S test with the same number of samples. However, the performance of the sequential K-S detector is very close to the batch K-S detector. Fig. 11 shows the average stopping time of the sequential K-S tests for different misbehavior strategies compared to the optimal SPRT at $P_D = 0.99$. The lines in the figure stop if the correspondent sequential K-S test is unable to obtain $P_D = 0.99$. Finally, we compare the average samples taken by the 1000-truncated sequential K-S test with the best batch K-S test, i.e., the fixed-sample-size K-S test that achieves the same P_D with the minimum number of samples. The figure shows that the sequential test is virtually as good as just selecting the best batch K-S test. This results shows that our sequential test does not have a significant performance loss compared to its batch

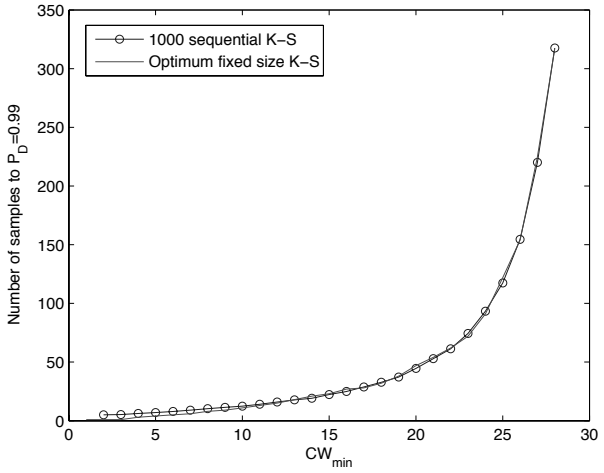


Fig. 12. Number of samples needed by the 1000-truncated sequential K-S vs. the optimum fixed size K-S.

counterpart, and hence it should be the preferred method.

VI. Conclusions

We have proposed a method for detecting misbehaving terminals in a CSMA/CA network, based on measuring the number of idle slots between successful transmissions. The Kolmogorov-Smirnov (K-S) test is employed to determine whether the samples are consistent with the hypothesis that the terminal abides by the protocol rules. We have proposed two detectors, a batch K-S detector and a sequential K-S detector, and we have applied them to detect misbehaviors in the IEEE 802.11 DCF protocol. The performance of the proposed non-parametric detectors is close to that of the optimum detectors that assume perfect knowledge about the misbehavior strategy. The proposed technique is, to the best of our knowledge, the first statistically robust misbehavior detector that can operate without modifying the protocol implementation.

References

[1] S. Radosavac, J. Baras, and I. Koutsopoulos, "A framework for MAC protocol misbehavior detection in wireless networks," in *WiSe '05: Proc. ACM Workshop on Wireless Security*, 2005, pp. 33–42.

[2] L. Buttyan and J. Hubaux, "Report on a working session on security in wireless ad hoc networks," *Mobile Computing and Communications Review*, vol. 6, no. 4, Nov. 2002.

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

[4] M. Cagalj, S. Ganeriwal, I. Aad, and J.-P. Hubaux, "On cheating in CSMA/CA ad hoc networks," in *Proc. 2005 IEEE Infocom*, Mar. 2005.

[5] 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, 2001.

[6] ——, "Stability of multipacket slotted aloha with selfish users and perfect information," in *Proc. 2003 IEEE Infocom*, Apr. 2003.

[7] A. Lopez 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.

[8] M. Raya, I. Aad, J.-P. Hubaux, and A. E. Fawal, "DOMINO: Detecting MAC layer greedy behavior in IEEE 802.11 hotspots," *IEEE Trans. Mobile Comput.*, vol. 5, no. 12, pp. 1691–1705, Dec. 2006.

[9] A. Cardenas, S. Radosavac, and J. S. Baras, "Detection and prevention of MAC layer misbehavior in ad hoc networks," in *SASN '04: Proc. ACM Workshop on Security of Ad Hoc Sensor Networks*, 2004, pp. 17–22.

[10] S. Buchegger and J. L. Boudec, "Performance analysis of the CONFIDANT protocol," in *Proc. MobiHoc'02*, June 2002, pp. 226–236.

[11] ——, "Nodes bearing grudges: Towards routing security, fairness, and robustness in mobile ad hoc networks," in *Proc. Euromicro Workshop on Parallel, Distributed and Network-Based Processing*, Jan. 2002, pp. 403–410.

[12] 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, 2004.

[13] H. Poor, *An Introduction to Signal Detection and Estimation*, 2nd ed. Springer-Verlag, 1994.

[14] A. Lopez Toledo and X. Wang, "A robust Kolmogorov-Smirnov detector for misbehavior in IEEE 802.11 DCF," in *Proc. IEEE Int'l Conf. Comm., (ICC'07)*, Glasgow, UK, June 2007.

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

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

[17] H. Khamis, "The δ-corrected kolmogorov-smirnov test for goodness of fit," *J. Statistical Planning and Inference*, vol. 24, pp. 317–335, 1990.

[18] ——, "The two-stage δ-corrected kolmogorov-smirnov test," *J. Applied Statistics*, vol. 27, no. 4, pp. 439–450, 2000.

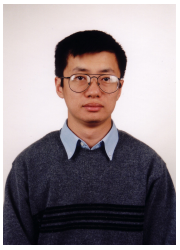
[19] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery, *Numerical Recipes in C: The Art of Scientific Computing*. New York, NY, USA: Cambridge University Press, 1992.

[20] S. McCanne and S. Floyd, network simulator 2. http://www.isi.edu/nsnam/ns.

[21] K. Medepalli, P. Gopalakrishnan, D. Famolari, and T. Kodama, "Voice capacity of ieee 802.11b, 802.11a and 802.11g wireless LANs," in *Proc. 2004 IEEE Globecom*, 2004, pp. 1459–1553.

Alberto Lopez Toledo (S'01) received the M.S. degree in Computer Science (with highest honors) from the University of Murcia (UMU), Spain, in 1999 and the M.S. degree in Electrical Engineering from Columbia University in 2003, where he is currently a Ph.D. candidate in the Electrical Engineering Department.

His research interests are in the area of wireless networking and cross-layer design. Mr. Lopez Toledo received Spains 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) received the Ph.D degree in Electrical Engineering from Princeton University. He is now on the faculty of the Department of Electrical Engineering, Columbia University. Dr. Wang's research interests fall in the general areas of computing, signal processing and communications, and has published extensively in these areas. Among his publications is a recent book entitled "Wireless Communication Systems: Advanced Techniques for Signal Reception", published by Prentice Hall in 2003. His current research interests include wireless communications, statistical signal processing, and genomic signal processing. Dr. Wang received the 1999 NSF CAREER Award, and the 2001 IEEE Communications Society and Information Theory Society Joint Paper Award. He has served as an Associate Editor for the *IEEE Transactions on Communications*, the *IEEE Transactions on Wireless Communications*, the *IEEE Transactions on Signal Processing*, and the *IEEE Transactions on Information Theory*.