Detecting MAC Layer Collision Abnormalities in CSMA/CA Wireless Networks

Alberto Lopez Toledo
Telefonica Research
alopezt@tid.es

Xiaodong Wang
Electrical Engineering Department,
Columbia University
wangx@ee.columbia.edu

***Abstract*—We present a robust non-parametric detection mechanism for CSMA/CA MAC 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 such observations. We show that the distribution of explainability of the collisions is very sensitive to abnormal changes in the network, even with a changing number competing terminals. NS-2 simulation results show that the proposed method has a very short detection latency and high detection accuracy.**

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 many nodes in a network [1]. While its correct operation assumes that all nodes obey the protocol, this may not be the case, as current wireless network devices have become easily programmable [2]. Such 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 or DoS attack).

MAC DoS attacks are particularly effective in wireless networks: they are stealthy by nature, as the attacker does not have to reveal itself in order to perform the attack, and more importantly, they require very little power [3], as only specific portions of other terminal's transmissions need to be targeted in order to succeed. The impact of such attacks on the network performance is often catastrophic: 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, so a terminal undergoing a few successive jams would virtually stop transmission. On the other hand, the random operation of the CSMA/CA protocol together with the nature of the wireless medium itself makes network conditions to appear different for different terminals [1]. Hence, it is difficult to determine if collisions are caused by link impairment, a surge in the number of terminals (such as in hot-spots), or by malicious terminals.

In this paper 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 to detect 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 non-parametric Kolmogorov-Smirnov detector.

II. Problem Formulation

Let $y_1, ..., y_K$ be a sequence of observations related to the state of a CSMA/CA network. We consider two hypotheses, hypothesis H_0 corresponding to the network performing normally, i.e., no jamming attack, and hypothesis H_1 corresponding to the case where the network is jammed. Hence we have the following hypothesis test:

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

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.

III. Statistical Analysis of Collisions under Normal Operation

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 [4] . In what follows we present an alternative characterization of the 'normal' operation of an IEEE 802.11 DCF network that a) does not depend directly on the observed collision probability; and b) is more sensitive to the presence of jammers than only tracking variations on the collision probability, and hence, offers improved detection capabilities.

A. The Contribution of a Terminal to a Collision

Consider an IEEE 802.11 DCF network with N competing terminals. We begin by analyzing the probability that each terminal in the network has participated in a collision, i.e., the probability that each terminal attempted a transmission in the slot corresponding to the collision.

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*,

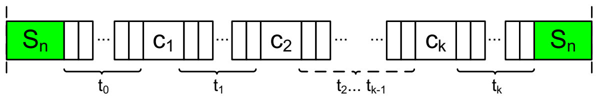


Fig. 1: The sequence of states in the network between two consecutive successful transmissions S_n of terminal n.

a *transmission slot*, or a *collision slot*. Consider the state of the network between two consecutive successful transmissions of terminal n. Because 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. 1. Assume that K collisions occur between the two transmissions of terminal n, denoted by $\{c_1, ..., c_K\}$, and define the corresponding *idle slot sequence* as $\{t_0, ..., t_K\}$, 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 $\{t_0, ..., t_K\}$.

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 $\{x_1^n, ..., x_K^n\}$ a *collision codeword*. We are interested in calculating the probabilities $p(x_i^n = 1|t_0, ..., t_K), i = 1, .., K, n = 1, .., N$. Denote W_i and w_i as the window size and the backoff counter[1] 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

$$p(x_1^n = 1|t_0) = P(u \leq t_0) = t_0/W_0. \quad (2)$$

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

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

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

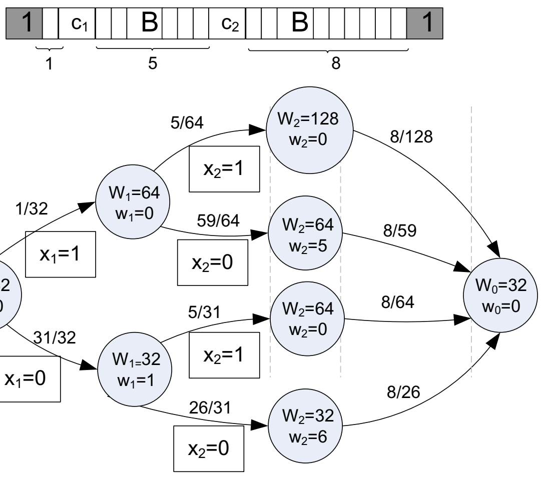


Fig. 2: Trellis corresponding to the idle slot sequence $\{t_0, t_1, t_2\} = \{1, 5, 8\}$.

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

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

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

We can use the above 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. 2 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 certain branch has probability 0, for example if $t_i > (W_{i-1} - w_{i-1})$.

From the above trellis we can compute the probability of each codeword for terminal n by multiplying the transition probabilities along the correspondent path in the trellis. Finally, we can estimate the probability of individual bits by marginalization as

$$p(x_i^n = 1|t_0, ..., t_K) = \sum_{x_i^n=1} p(x_1^n, ..., x_K^n|t_0, ..., t_K). \quad (5)$$

The computation of the marginal probabilities in (5) can be efficiently implemented by using the *forward-backward algorithm* [5].

Note that the total number of nodes at the K-th 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[2] ranging from 1 to 25, we observe from ns-2 simulations that the chance of having $K > 20$ is only about 0.9%. The forward-backward algorithm takes less than 3 seconds in decoding codewords with length 20, and 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 (5) may become intractable. In those cases, a simple Monte Carlo approximation can be used.

Algorithm 1 Calculation of collision explainability

```
1: Observe the network until a successful transmission from any terminal is observed. Let that be a transmission from terminal $n \in \{1, ..., N\}$.
2: Denote $\{c_1, c_2, ..., c_C\}$ as the sequence of collisions and $\{t_0, ..., t_K\}$ the sequence of idle slots observed in the network since the last successful transmission of terminal $n$.
3: Calculate the contribution of $n$ to the collisions $\{c_1, c_2, ..., c_C\}$, i.e., $p(x_1^n|t_0, ..., t_K), ..., p(x_C^n|t_0, ..., t_K)$.
4: For those collisions $c_i \in \{c_1, c_2, ..., c_C\}$ for which all the $p(x_i^j|\cdot), j = 1, ..., N$ are known (where $N$ is the number of competing terminals in the network), calculate $e(c_i)$ given by (7).
5: Go to Step 1.
```

B. The 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 $\{c_1, c_2, ..., c_C\}$ be the sequence of collisions in the network over a period of time. We can calculate the probability $p(x_i^n|t_0, ..., t_K)$ that each terminal n contributed to each of the collisions c_i, where $\{t_0, ..., t_K\}$ refers to the idle slot sequence relevant[3] to collision c_i. When there is no jammer present in the network, a collision event is defined as

$$c_i = \mathbb{1}\left\{\sum_{i=1}^{N} x_i^n \geq 2\right\}, \tag{6}$$

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

$$\begin{aligned} e(c_i) &\triangleq \mathbb{E}\left\{\mathbb{1}\left\{\sum_{i=1}^{N} x_i^n \geq 2.\right\}\Big| t_0, ..., t_K\right\} \\ &= P\left(\sum_{n=1}^{N} x_i^n \geq 2 \Big| t_0, ..., t_K\right) \\ &= 1 - \prod_{n=1}^{N} p(x_i^n = 0|t_0, ..., t_K) \\ &\quad - \sum_{n=1}^{N} p(x_i^n = 1|t_0, ..., t_K) \prod_{j \neq n} p(x_i^j = 0|t_0, ..., t_K), \end{aligned} \tag{7}$$

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. 3. For each terminal n in the network, we calculate the probabilities $p(x_i^n|t_0, ..., t_K)$ that the terminal had participated in collision c_i 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 the probabilities $p(x_i^n|t_0, ..., t_K), n = 1, ..., N$ are known, we use (7) to calculate its explainability $e(c_i)$. The process is detailed in Algorithm 1. 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.

Figs. 4(a) and 4(b) show the cumulative distribution functions (cdf) of $e(c_i)$ obtained by Algorithm 1 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 the collisions can be explained with 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 $e(c_i) = 1$.

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 (7) would suffice to determine whether a collision is caused by a jammer or not. On the other hand, and under normal protocol operation, the explainabilities $e(c_i)$ have the distribution shown in Figs. 4(a) and 4(b), and some collisions can be explained better than others. More importantly, the *distribution* of $e(c_i)$ is an excellent indicator of the normal

[2]Note that 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.

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

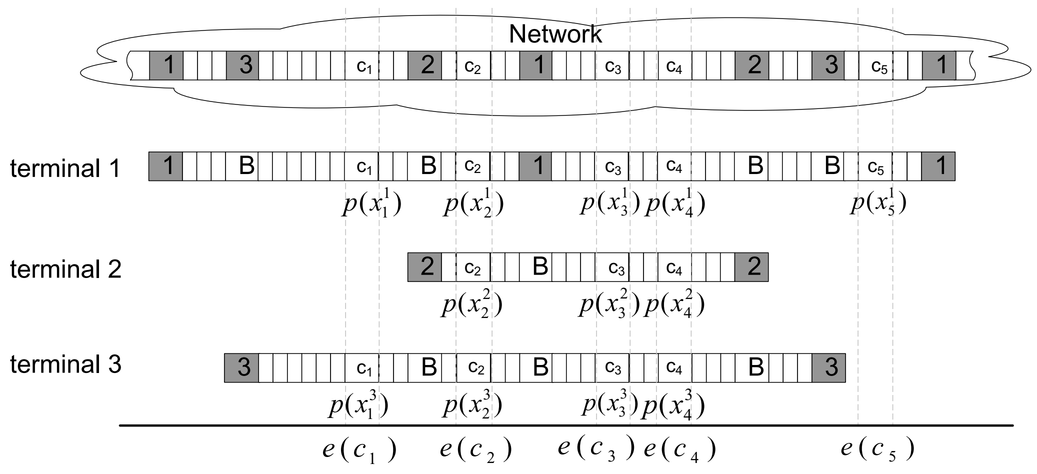
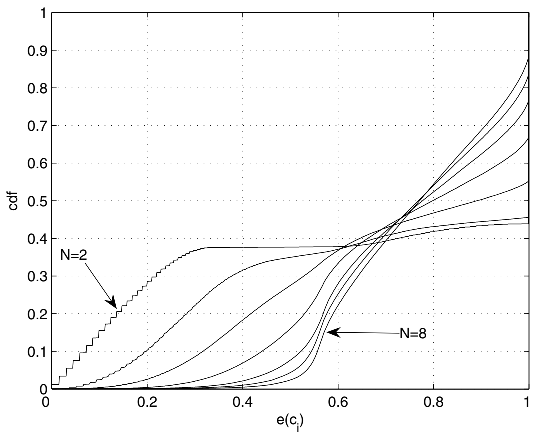
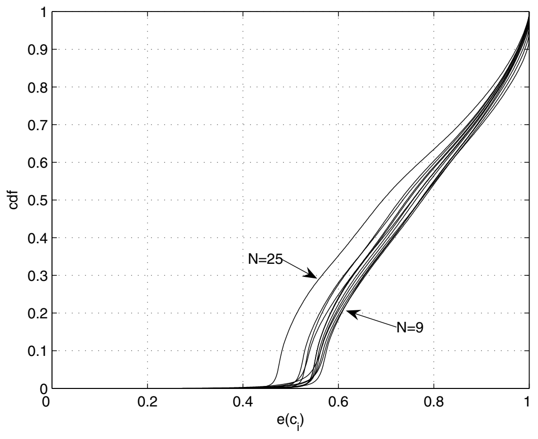


Fig. 3: Example of calculating $e(c_i)$. The conditionals on the relevant idle slot sequences $\{t_0, ..., t_K\}$ are not shown for brevity.

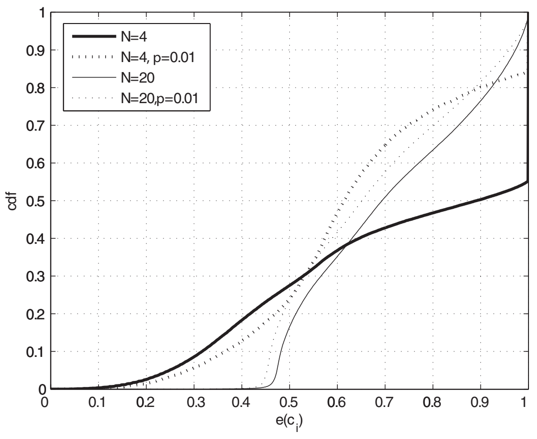


(a) For the number of competing terminals between 2 and 8.

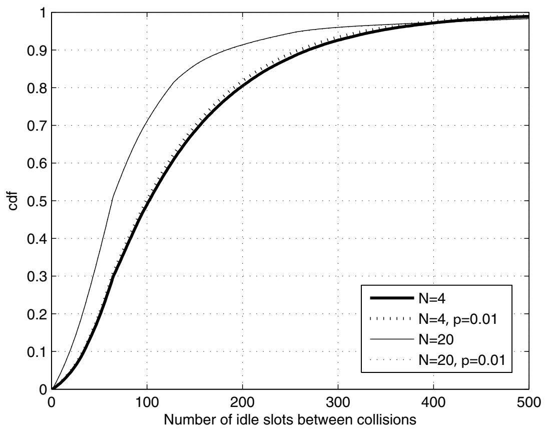


(b) For the number of competing terminals between 9 and 20.

Fig. 4: Distribution of the explainability of the collisions $e(c_i)$ in IEEE 802.11 DCF calculated using Algorithm 1.



(a) cdf of collision explainability $e(c_i)$.



(b) cdf of the number of idle slots between collisions.

Fig. 5: Effect of a jammer in the network.

protocol operation for a given number of competing terminals. Fig. 5(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. The first is that the cdf of $e(c_i)$ may significantly change its shape in the presence of a jammer. The second point is that the cdf of the explainability of collisions change dramatically even for a very small percentage of corrupted frames as we can see in Fig. 5(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. 5(b), does not experience significant change under light jamming, and it 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. The 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 detection of a jamming attack. However, Algorithm 1 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. Note that because a jammer artificially affects the observed probability of collisions, we can not use other techniques to track the number of competing terminals such as [6] [7], that assume that the network is performing normally.

A. Probability that a Terminal is Still Competing

Here 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.

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 $CW_{\min} = 32$ and $CW_{\max} = 1024$, T can be analytically computed. Let $\hat{p}_c^{(1)}, ..., \hat{p}_c^{(q)}$ be a sequence of collision probability estimates obtained by

$$\hat{p}_c = \frac{\text{\# of collisions}}{\text{\# of transmissions}}. \tag{8}$$

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

$$\begin{aligned} P_{\mathcal{C}}^n(t) &= P(\mathcal{C}|T_n > t) \\ &= \frac{P(T_n > t|\mathcal{C})P(\mathcal{C})}{P(T_n > t|\mathcal{C})P(\mathcal{C}) + P(T_n > t|\overline{\mathcal{C}})P(\overline{\mathcal{C}})}, \end{aligned} \tag{9}$$

where $\mathcal{C}$ is the event that a terminal is still competing, and $\overline{\mathcal{C}}$ is the event that a terminal has ceased to compete. Note that $P(T_n > t|\mathcal{C}) = 1 - \hat{G}_{idle}(t)$, where $\hat{G}_{idle}(t)$ is the estimated cdf of T. Also note that if terminal n is not competing, then $P(T_n > t|\overline{\mathcal{C}}) = 1$. The probability $P(\mathcal{C})$, i.e., the overall probability that after a successful transmission a terminal still has data to send, depends on both 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 $\sum_{i=\tau}^{T_{\max}} P_{\mathcal{C}}^n(i) = \eta$, η is the desired false alarm probability, and $T_{\max} = \max(T)$.

We assume that the arrival and departure rate of a terminal in the network happens 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 $\eta = 0.01$, 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 above 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 $p(x_i^n|t_0, ..., t_K)$ 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 than $T_{\max}$.

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

B. Sequential DoS Detector

As we discussed in Section II, we are interested in developing a detector that can discriminate between a 'normal' operation of the network characterized by a probability distribution f_0, and an '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 $\{e(c_1), ..., e(c_K)\} \sim f_i, i = 0, 1$. Because 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 test introduced in [1].

Algorithm 2 Detecting jammer attacks using the M-truncated sequential K-S test with $P_{FA} = \alpha$

```
1:  $m = 0$.
2:  $\beta \leftarrow 1 - \sqrt[M]{1-\alpha}$.
3:  $m \leftarrow m + 1$.
4:  Let $\{e(c_i), ..., e(c_{i+C})\}$ be last values returned by Algorithm 1, and $\{N(c_i), ..., N(c_{i+C})\}$ 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 $\{N(c_i), ..., N(c_{i+C})\}$ using (11).
6:  Update the edf $F_1$ with the observations $\{e(c_i), ..., e(c_{i+C})\}$ using (12).
7:  Calculate the significance level $P$ of the stage as in (16).
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
```

The Kolmogorov-Smirnov (K-S) test [8], 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) [9]. The K-S test compares the edf F_1 obtained from the data samples with the hypothesized cdf F_0, and determines whether $F_1 = F_0$, or $F_1 < F_0$, or $F_1 > F_0$. For the jamming detection problem, we use the following test

$$\text{choose} \begin{cases} H_0 : F_1 = F_0 & \text{(no jamming)} \\ H_1 : F_1 \neq F_0 & \text{(jamming).} \end{cases} \tag{10}$$

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 (cf. Fig. 4). Then, for a given sequence of collisions in the network $\{c_1, ..., c_C\}$, the distribution F_0 for the test in (10) is given by

$$F_0 = \frac{1}{C} \sum_{i=1}^{C} F_0^{N(c_i)}, \tag{11}$$

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. Also let $\{e(c_1), ..., e(c_C)\}$ be the corresponding sequence of explainability of collisions. Then, the edf F_1 of the observations for the test in (10) is given by

$$F_1(e(c_j)) = \frac{1}{C} \sum_{i=1}^{C} \mathbb{1} \{e(c_i) \leq e(c_j)\}. \tag{12}$$

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

$$\hat{D} = \max_{1 \leq i \leq C} \{F_1(e(c_i)) - F_0(e(c_i))\}. \tag{13}$$

Define

$$\lambda(\hat{D}) = \max\left\{ \left(\sqrt{C} + 0.12 + \frac{0.11}{\sqrt{C}} \right) \hat{D}, 0 \right\}, \tag{14}$$

$$\beta = 1 - \sqrt[M]{1-\alpha}, \tag{15}$$

where C is the number of samples (i.e., collisions), and M is the maximum stage the M-truncated sequential K-S test with 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 [10]

$$P = e^{-2\lambda(\hat{D})^2}. \tag{16}$$

Finally, the algorithm for detecting the presence of a jammer in the network is summarized in Algorithm 2.

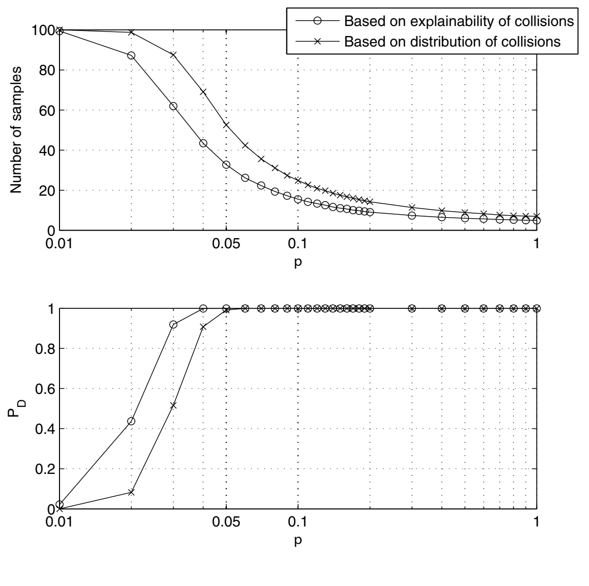
V. Simulation Results

A. Simulation Setup

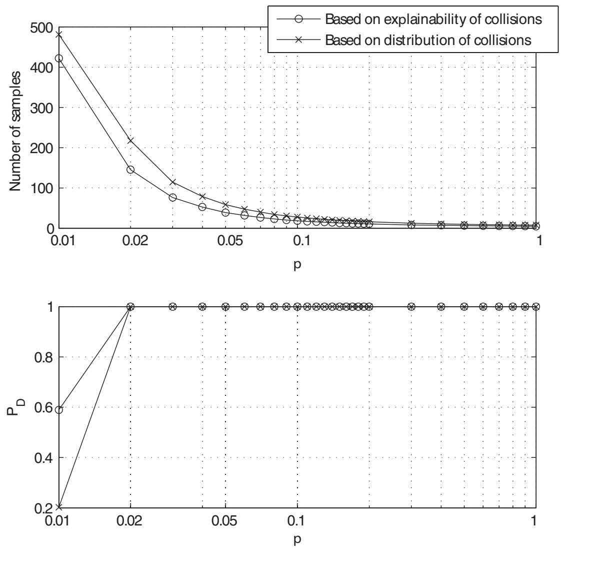
We consider the IEEE 802.11 DCF where a legitimate terminal uses $CW_{\min} = 32$ and $CW_{\max} = 1024$. For all the experiments in this paper, as well as for the figures in the previous sections, data are collected using the ns-2 network simulator version 2.28 [11], and we implement the detection algorithm using MATLAB. 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 the other terminals and implements the detection algorithm. 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. 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 seconds and idle time mean (off) of 5 seconds.

We only consider the case of intelligent jamming [12], i.e., the jammer corrupts frames with the knowledge of the protocol. For simplicity, we consider *p-random jamming* attacks, where the jammer corrupts the CTS frames in the network with probability p. A particular case of this attack is the *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.

For comparison, apart from the detector described in Algorithm 2, 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



(a) 100-truncated K-S.



(b) 500-truncated K-S.

Fig. 6: Number of samples and probability of detection (P_D) of the detectors when a p-random jammer is present and $P_{FA} = 0.01$.

is better suited for this problem than detectors based on the frequency or probability of collisions.

B. Results

Fig. 6 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 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 milliseconds in a standard IEEE 802.11b network.

VI. Conclusions

We have proposed a method for detecting MAC layer denial-of-service (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 Kolmogorov-Smirnov (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 both 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, as it is able to detect any deviation from the 'normal' operation of the network, and can operates without modifying the protocol implementation.

References

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

[2] S. Radosavac, J. Baras, and I. Koutsopoulos, "A framework for MAC protocol misbehavior detection in wireless networks," in *WiSe '05: Proc. ACM Workshop Wireless Security*, Cologne, Germany, Sep. 2005.

[3] M. Acharya, T. Sharma, D. Thuente, and D. Sizemore, "Intelligent jamming in 802.11b wireless networks," in *Proc. OPNETWORK 2004*, Washington, DC, Aug. 2004.

[4] 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.

[5] D. Mackay, *Information Theory, Inference & Learning Algorithms*. Cambridge, UK: Cambridge University Press, 2003.

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

[7] 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 Process.*, vol. 55, no. 2, pp. 437–450, Feb. 2007.

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

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

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

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

[12] D. Thuente and M. Acharya, "Intelligent jamming in wireless networks with applcations to 802.11b and other networks," in *Proc. 2006 IEEE MILCOM*, Washington, DC, Oct. 2006.