Detecting MAC Layer Back-off Timer Violations in Mobile Ad Hoc Networks

Venkata Nishanth Lolla ‡, Lap Kong Law ‡, Srikanth V. Krishnamurthy ‡,
Chinya Ravishankar ‡, and Dharmaiah Manjunath †

‡Department of Computer Science & Engineering, University of California, Riverside

†Department of Electrical Engineering, Indian Institute of Technology - Mumbai

{vlolla,lklaw,krish,ravi}@cs.ucr.edu, dmanju@ee.iitb.ac.in

Abstract

In IEEE 802.11 based ad hoc networks, by simply manipulating the back-off timers and/or wait times prior to transmission, malicious nodes can cause a drastically reduced allocation of bandwidth to well-behaved nodes. This can result in causing bandwidth starvation and hence, a denial of service to legitimate nodes. We propose a combination of deterministic and statistical methods that facilitate detection of such misbehavior. With our approach, each of the nodes is made aware of the pseudo-random sequences that dictate the back-off times of all its one-hop neighbors. A blatant violation of the timer is thus, immediately detected. In certain cases, a node may be unable to monitor the activities of its neighbor and therefore deterministically ascertain if the neighbor is misbehaving. To cope with such cases, we propose a statistical inference method, wherein based on an auto-regressive moving average (ARMA) of observations of the system state, a node is able to estimate if its neighbor is indulging in misbehavior. Simulation results show that with our methods, it is possible to detect a malicious node with a probability close to one. Furthermore, the probability of false alarms is lower than 1%.

1 Introduction

The IEEE 802.11 MAC protocol (or variants thereof) has been popularly considered for use in ad hoc networks. The decentralized random access nature of this protocol makes it especially vulnerable to attacks. Compromised malicious nodes can violate protocol rules to present operational problems and to cause denial of service to legitimate and well-behaved nodes in the network[1]. A compromised malicious node could launch a subtle attack wherein it would simply violate the back-off timer specifications of the IEEE 802.11 standard. The distributed operational framework of medium access control and the lack of a centralized arbiter (such as an access point) makes such timer violations especially hard to detect. In this paper, we design a framework for detecting such attacks and identifying the attackers.

A node is deemed to be malicious or simply misbehaving if it does not adhere to the IEEE 802.11 MAC. By using smaller timeouts than that specified in the protocol standard, especially at high loads, a misbehaving node can gain an unfair advantage by acquiring the wireless channel more often than its legitimate and well-behaved neighbor nodes. With IEEE 802.11, each node is expected to choose a back-off interval prior to initiating a transmission. The back-off interval is to be increased as per a specific set of rules prior to retransmission attempts that are invoked upon failed transmission attempts[2] [1]. A malicious node may choose a small and/or a constant back-off interval prior to the transmission of a data packet or follow a completely different retransmission strategy upon experiencing failed transmissions that does not conform to the standard IEEE 802.11 protocol rules. This in turn would prevent media access by the node's neighbors. The properties of IEEE 802.11 would magnify this effect and would result in a denial of service to these neighbor nodes. The detection of such attacks and the identification of the misbehaving attackers is thus a critical problem and is addressed in this paper.

We propose a novel framework that relies on both deterministic and statistical methods to detect medium access misbehavior resulting from time-out violations. Our detection mechanism discourages, and thereby can be expected to prevent misbehavior to a significant extent. The key ideas in our approach include the following: (a) Each node is required to include information with regard to the state of its random number generator that governs its back-off when transmitting a RTS (request to send) or CTS (clear to send) message. (b) Each node uses a statistical inference framework wherein it monitors the transmission patterns of its

[1] Compromised nodes are previously legitimate participant nodes that have been taken over by an attacker.

[2] To be discussed in detail later.

neighbors and hypothesizes if a particular neighbor is not adhering to proper back-off semantics.

The statistical estimations are needed since a monitoring node may at times perceive a *system state* that is not consistent with the *system state* of the particular neighbor that it is monitoring. Here, *the system state* of a node refers to the offered traffic intensity in its vicinity and the density of nodes in its local neighborhood. In some cases, a node would be able to deterministically ascertain if a neighbor is allowed to count down its back-off timer. However, in certain other scenarios, a node may not be able to deterministically estimate if and when a neighbor's back-off timer is frozen due to the latter being within the interference footprint of a third party node. We will discuss these issues in detail later.

We develop a simple analytical model that allows each node to statistically estimate if a neighbor is in compliance with the back-off rules specified by the IEEE 802.11 standard. We evaluate our framework by means of extensive simulations wherein we consider more realistic, heterogeneous traffic and channel effects due to pathloss and shadowing. We find that, each node, by monitoring the behavior of its neighbors over reasonably short periods of time, can detect if any of these neighbors are in violation of the timers. Our framework also ensures that the probability of false assessment of misbehavior is extremely small (less than 0.01).

The rest of the paper is organized as follows. In Section 2, we present the necessary background for our work and discuss the related work in brief. In Section 3, we discuss the problems associated with misbehavior detection in ad hoc networks and deliberate on the need for statistical methods. The schemes that we propose in order to detect malicious behavior of nodes in an ad hoc network and the associated analytical models form Section 4. In Section 5, we present results from our simulations, and discuss the results in detail. Our final section summarizes our conclusions and presents our thoughts for future possibilities.

2 Background and Related Work

In this section, we first briefly describe the IEEE 802.11 back-off algorithm since this is key to the development of the rest of our paper. Next, we provide a description of prior work on security in ad hoc networks in brief.

The IEEE 802.11 Back-Off Algorithm: As in popular prior art [2, 4, 6, 13, 14], we assume that the distributed coordination function (DCF) of the IEEE 802.11 MAC is used. The DCF is based on carrier sense multiple access with collision avoidance (CSMA/CA) for resolving contention among multiple nodes. The collision avoidance of CSMA/CA is supported via a random back-off procedure. We wish to point out here that most of the analytical formulations of the IEEE 802.11 MAC protocol (as in [2]) do not accurately model the hidden terminal or the interference range effects. In this paper (as we discuss later) we construct a simple approximate model that we validate via simulations.

In IEEE 802.11, time is slotted into units of time-slots and is used to define the inter-frame-space (IFS) intervals and to determine the back-off times for contending nodes in the network. If a node, with a packet to transmit, initially senses the channel to be busy, it waits until the channel becomes idle for a Distributed Inter Frame Space (DIFS) period, and then computes a random back-off time. The random back-off time is specified by an integer value that corresponds to a number of time slots. The idle period after a DIFS period is referred to as the contention window (CW). Initially, the node under discussion computes a back-off time in the range [0,CW_{min}], where CW_{min} is the minimum contention window size. When the medium becomes idle, after an additional DIFS period, nodes decrement their back-off timers until the medium becomes busy again or until the timer value reaches zero. If the timer has not reached zero and the medium becomes busy, the node *freezes* its timer. This procedure continues until the timer is finally decremented to zero. Then, the node transmits its packet. If two or more nodes decrement their timers to zero at the same time, a collision will occur, and each node will have to generate a new back-off time that would now lie in the range [0,2*CW_{min}]. In this manner, the back-off time is selected randomly from the range [0,2^i*CW_{min}] during the i^{th} retransmission attempt. This provides a means of avoiding repeated collisions when there is congestion.

Related Work: The ad hoc network community has tried to understand and address issues related to attack resistance in recent times [23, 26]. One of the recent focal areas of research has been on thwarting routing attacks [7, 8, 10, 14, 21]. In [14], the authors propose the use of watchdogs to ensure that nodes do not misbehave while forwarding packets of other nodes. In [22], the authors suggest the AD-MIX protocol to discourage selfishness by nodes in terms of forwarding data packets. Concealing the true destination of a packet from intermediate nodes encourages data forwarding; this forces a node to participate or risk dropping packets that may be destined for the node itself. Zapata and Asokan demonstrated that, via an advertisement of a route with a smaller distance metric than the actual distance to the destination or by means of sending routing updates with large sequence numbers, a malicious node could invalidate all routing updates from legitimate nodes [25].

There have been prior studies that examine DoS attacks at the MAC layer [6]; however, these studies do not propose any concrete solutions. There has been some work on detecting MAC layer misbehavior in Wireless LANs. In [13], a scheme in which the access point (AP) selects a random back-off value for each sender and piggybacks this selection on to the CTS and ACK packets, is proposed. The mobile

sender has to use the back-off value specified by the *trusted* receiver prior to its next transmission. Our work differes from the work in [13] in the following ways: (i) Our framework solves the problem of misbehavior in the absence of a trusted *centralized* AP in ad hoc environments. (ii) We consider the effect of interference/sensing range (neglected in all previous work) of a node in analyzing and estimating the *system state* (to be discussed).

Raya et al. [17] present a system, which is implemented at the AP to detect the misbehavior of wireless nodes at the MAC layer. Traffic traces, collected regularly, are passed through a certain series of tests to detect greedy access behavior. The method although viable, will introduce large delays in detecting misbehavior.

Protocols based on game-theoretic techniques [11, 12, 16] have also been designed to provide resilience to misbehavior. Konorski designed a modified back-off algorithm at the MAC layer that can endure malicious behavior in wireless LANs [11, 12]; in contrast, we consider a generalized *multi-hop* ad hoc network. Michiardi et al. [16] model the nodes in the network as participants in a non-cooperative game with each node attempting to maximize its own *utility*.

3 The System Model

In this section, we provide a description of the problem that we consider and discuss the need for statistical estimates of system state in order to detect back-off timer violations. During the discussion, we also highlight the various assumptions/approximations that we make and their impact on our overall framework.

The Problem: The problem that we consider in this paper is the one of detecting the misbehavior (possibly with malicious intent) of nodes in terms of violating the back-off timers associated with the IEEE 802.11 MAC protocol in an ad hoc network setting. Misbehaving nodes may choose a shorter back-off time possibly generated via a distribution different from that with the IEEE 802.11 MAC or simply follow a different retransmission strategy. Such misbehavior could also deny legitimate channel access to the other nodes in the network. The nodes in an ad hoc network cannot rely on a single trusted entity to monitor such misbehavior. Therefore, each of the nodes will have to monitor its neighbors to detect violations of the access policies.

Using Verifiable Back-off timers: This problem identifies the need for a deterministic/known sequence of back-off values that each node will have to follow. The existence of such a globally known sequence will allow the neighbors of a node to monitor its back-off times and detect misbehavior in a large number of scenarios. In some scenarios, however, even with this exchange, it might not be possible for nodes to deterministically ascertain the misbehavior of a neighbor. But, dictating such a known sequence of random back-off values for each node has two additional advantages:

- It discourages nodes, now aware of the protocol semantics, from indulging in such misbehavior.
- In scenarios where there is some uncertainty with regard to whether a node's neighbor is misbehaving, it is easy for the node to statistically estimate the probability of the neighbor's misbehavior based on observed patterns.

We wish to state here that we do not address the problem of MAC address spoofing in this paper. Prior work on the design of certificate authorities for effectively using public key infrastructure in ad hoc networks [24] are assumed to sufficiently thwart attempts to spoof MAC addresses.

Making Sense of the Uncertainty in System State: To diagnose the misbehavior of a neighbor correctly, it is important for the monitoring node to estimate appropriately, the *system state* of its neighbors.

For simplicity, we explain the scenario with two neighboring nodes namely S, the sender, and R, the receiver as shown in Figure 1. Let us assume that R is monitoring S. A similar example may be easily constructed wherein, as opposed to being the receiver of information from S, R is simply a third-party neighbor of S. Although S and R share a common wireless channel, the *system state* observed by S and R might differ. For example, S might sense the channel to be busy due to interference from nodes within its sensing range (example from T) while the receiver senses it to be idle and vice versa[3]. So, a run time estimation of the *system state* would be needed for R to estimate the number of busy/idle slots observed by S when R itself is either (a) idle or (b) busy due to a transmission in its sensing range (such as V). In these situations, the only knowledge that the receiver/monitor can potentially have (if a monitoring process is in place) is the number of idle (I) / busy (B) slots that it observes within a chosen observation period of N time slots (the sample interval). Note that $I + B = N$.

Even though the load experienced by different nodes are likely to be different, one could still expect that the loads experienced by nodes in the vicinity of each other would be of the same order (in other words, approximately equivalent). This is because the interference projections span a fairly large area (the interference region is a unit disk of radius 550m in typical models used with the IEEE 802.11 MAC protocol) and hence, nodes that are in the vicinity of each other, are likely to be in similar interference zones. *Thus, in our framework, a node assumes that the steady state loads experienced by all the nodes within its two hop radius, are identical.* While this assumption makes the estimate of the

[3]Note that the nodes that are said to be in the transmission range of a node can decode the information sent by the node. Nodes that are in the sensing range of the node simply sense the physical medium to be busy due to the transmission by the node [1].

sender's system state approximate, it does not require nodes to exchange load information using which, a better estimate can be made. Based on the above homogeneity assumption, with the observed values of I and B, the monitor (R) can approximately estimate the number of idle (I_{est}) and busy (B_{est}) slots from the sender's (S's) perspective using the following equations:

$$I_{est} = P_{I/I} * I + P_{I/B} * B \quad (1)$$

$$B_{est} = N - I_{est} \quad (2)$$

where $P_{I/I}$ is the probability that sender (S) senses the shared channel to be idle, given that the monitor/receiver (R) senses the channel to be idle and $P_{I/B}$ is the probability that the sender (S) senses the shared channel to be idle, given that the monitor (R) senses it to be busy. The known factors in the equations are *N*, *I* and *B* whereas $P_{I/I}$ and $P_{I/B}$ are unknown. Essentially, our goal is to compute these probabilities by taking into account a fairly realistic two-hop neighborhood. Note here that this model takes into account the interference effects from within the sensing range and the effect of hidden terminals. We define certain areas (A_1, A_2, A_3, A_4 and A_5) as shown in Figure 1. These areas represent portions of the sensing range of the nodes of interest, viz. S and R.

Derivation of $P_{B/I}$: Let the number of nodes in region A_1 be *k* and the number of nodes in region A_2 be *n*. From Figure 1, if the monitor (R) senses the channel to be idle during a slot, it would mean that no other node that is in its sensing region is transmitting at the time of sensing. In other words, no nodes that are in areas A_3 or A_4 or A_5 are transmitting at this time. For S to be busy during this slot, at least one of the *n* nodes from region A_2 should be transmitting. Although more than one node in A_2 could be potentially transmitting, the probability of this event would be negligible, since in most instances, the transmission of one node would cause the channel to be busy from the perspective of all the other nodes in A_2. Thus, we assume that only one node in A_2 is involved in a transmission if S were to sense the channel to be busy.

Depending on the position of this transmitting node in A_2, certain nodes in region A_1 will be precluded from simultaneous transmissions. This is a consequence of the fact that, for the node to transmit, other nodes in its sensing range should be idle. We make an assumption that with a transmission in A_2, all nodes in A_1 are precluded from transmissions and vice versa. This makes our analysis approximate; however, as we shall show later, the results more or less concur with actual observations (as it pertains to the desired probability) in simulations. Given this assumption, and that there is exactly one node from among the nodes in ($A_1 \bigcup A_2$) transmitting during the slot, the probability that the transmitting node is from A_2 is simply $\frac{A_2}{A_1+A_2}$[4].

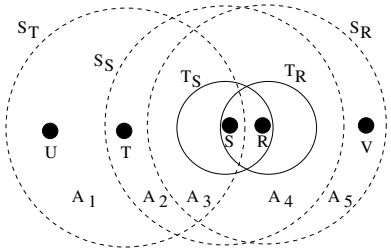


Figure 1. S_X and T_X are Sensing & Transmission Range of X, respectively. U, T, S, R and V are wireless nodes placed at various positions. A_1, A_2, A_3, A_4 and A_5 represent the area of regions enclosed between their respective left and right arcs. A circle with dotted line represents the sensing range and a circle with solid line represents the transmission range.

Next, we compute the probability that at least one of the nodes in ($A_1 \bigcup A_2$) is transmitting. Here, we make a second approximation by ignoring the effects from transmissions beyond the area of interest i.e., beyond ($A_1 \bigcup A_2 \bigcup A_3 \bigcup A_4 \bigcup A_5$). With this assumption, if ρ is the traffic intensity (ratio of the arrival rate to the service rate of MAC layer packets) observed by the monitoring node, the computed steady state probability that any chosen node in ($A_1 \bigcup A_2$) has an empty queue is simply given by (1-ρ) [9][5]. In order to make the analytical computation tractable, we make a third assumption i.e., the queues are independent[6]. With this, the probability that none of the (*n*+*k*) nodes have packets to transmit is then given by $(1-\rho)^{n+k}$. Thus, the probability that a node in ($A_1 \bigcup A_2$) transmits is $\{1-(1-\rho)^{n+k}\}$. Hence, the desired probability $P_{B/I}$ is the product of the conditional probability $P(\frac{\text{The transmitting node is in } A_2}{\text{There is a node transmitting in } (A_1 \bigcup A_2)})$ and the probability of a node transmitting in ($A_1 \bigcup A_2$) and is thus computed as:

$$P_{B/I} = \{\frac{A_2}{A_1 + A_2}\} * \{1 - (1-\rho)^{n+k}\} \quad (3)$$

Derivation of $P_{I/B}$: The computation of $P_{I/B}$ is similar to the previous computation of $P_{B/I}$. This event would imply that there are no nodes in A_3 that are transmitting at this time. Let there be *m* nodes in region A_4 and *j* nodes in region A_5. Again, from Figure 1, if the monitor (R) senses the channel to be busy during a particular time slot, it means

[4]Note that this is a consequence of the assumption that nodes are uniformly distributed in the area of interest.

[5]Note here that the routing artifacts and multi-hop behaviors are already accounted for, when the monitoring node computes the traffic intensity (ρ) online.

[6]Our simulations demonstrate that even with this assumption, the computed probability is similar to the real probability.

that some node that is within its sensing region is transmitting a packet. For S to sense the channel to be idle at this time, the transmission could only be from a node in region A_5. As in our previous analysis, assuming that only a single node in $(A_4 \bigcup A_5)$ can be transmitting at any given time, the probability that the transmitting node (out of *m+j* nodes) comes from region A_5 is simply $\frac{A_5}{A_4+A_5}$. Given that this is the case, it is possible for a potential transmission to occur in $(A_1 \bigcup A_2)$. If no transmissions occur in this area, then S is idle. If a transmission were to occur, in order for S to sense the channel to be idle, this transmission should occur from one of the *k* nodes in region A_1 and not from one of the *n* nodes in A_2. Thus, the probability that S would sense the channel to be idle is computed to be:

$$\{\frac{A_1}{A_1+A_2} * [1-(1-\rho)^{n+k}]\} + (1-\rho)^{n+k}$$

The first term accounts for the case wherein given that a transmission occurs in $(A_1 \bigcup A_2)$, the transmitting node belongs to A_1. The second term accounts for the case wherein all of the queues in $(A_1 \bigcup A_2)$ are empty and no transmissions occur from among the nodes in this region. Combining the above expressions, we compute $P_{I/B}$ as follows:

$$P_{I/B} = \{\frac{A_5}{A_4+A_5}\}*\{(\frac{A_1}{A_1+A_2})*[1-(1-\rho)^{n+k}]+(1-\rho)^{n+k}\} \quad (4)$$

Since $P_{I/I} + P_{B/I} = 1$,

$$P_{I/I} = 1 - P_{B/I} \quad (5)$$

The results of Equations 3, 4 and 5 are used by monitoring nodes to estimate the number of idle and busy slots perceived to be seen by the sender as specified in Equations 1 and 2. We reiterate that, we find via simulations that in spite of our simplifying approximations, our estimates are fairly accurate in predicting the misbehavior of nodes.

4 Our proposed framework

In this section, we describe our proposed framework for detecting MAC layer timer violations in the ad hoc network.

Overview of the approach: Let us assume that there is a node being monitored (we call this the tagged node) and consider a particular monitoring neighbor of the tagged node. In a nutshell, our framework functions as follows:

- The tagged node announces (as per our requirement) the state of its pseudo-random sequence generator using which, the monitoring neighbor can determine the sequence of back-off times to be used by the tagged node.
- The monitoring neighbor, in some cases, may not be able to deterministically determine if the tagged node is using a legitimate back-off countdown process. In such cases, the observed back-off times would differ from what the monitoring node computes using the announced pseudo-random sequence generator state (this computed back-off time is the expected back-off time).
- The monitoring neighbor, then uses a hypothesis test (specifically the Wilcoxon rank sum test) based on its online estimates of the probabilities $P_{I/B}$ and $P_{B/I}$ (described in the previous section), to determine if the difference between the observed and expected back-off times is sufficient to deem the tagged node as a misbehaving node.

Remark: In the proposed framework, one may argue that a node can examine the back-off timers announced by its neighboring nodes and may then try to set its back-off time to be slightly smaller than the announced values to win the contention phase. However, such an action can be easily detected since the monitoring node is also monitored by its neighbors in a similar fashion.

Online estimation of system state: As described in the previous section, the **traffic intensity** (ρ) experienced by a monitoring node is computed based on the number of busy/idle slots observed on the channel. A slot is categorized as busy (idle) if the monitor senses the channel to be busy (idle) during that particular slot. In effect, we define the traffic intensity as the fraction of the total number of busy slots (*B*) out of the *N* observed slots. i.e., $\rho = \frac{B}{N}$. Note that this definition is consistent with our previous definition of ρ in Section 3. The run-time estimation of the traffic intensity is provided by a simple mechanism by using an ARMA filter [3]. In particular, we use the following:

$$\rho(t+1) = \alpha * \rho(t) + (1-\alpha) * \frac{1}{s}\sum_{i=0}^{s-1} C_{t-i} \quad (6)$$

where *s* is the sample size. $C_i = 0$ (or $C_i = 1$) if the node senses the channel to be idle (or busy) during the i^{th} slot. ρ(t) is the ARMA smoothing of the traffic intensity experienced by the monitoring node with a moving average taken over the last *s* samples. The parameter α is taken to be .995 as in previous systems that use this method for online estimations [3]. We find that our results are not very sensitive to the value of α, as long as α is close to 1.

In [3], Bianchi and Tinnirello express the number of competing terminals as a function of the collision probability encountered on the wireless channel and then, estimate this number based on run-time measurements. On estimating the number of competing terminals within its vicinity, a monitoring node can approximate the **network density** in its extended neighborhood (transmission/sensing range and further beyond). Let the transmission range be R and the number thus computed by a particular monitoring node be n_T. Then, the number of nodes in a given area A_j is simply estimated to be $\frac{n_T}{\pi R^2} \times A_j$. Note that this estimation is valid only if the network has a uniform distribution of nodes.

In non-uniform densities (not considered in this work) one may require explicit verifiable reports from nodes with regard to their degrees in order to enable monitors compute the node density that is perceived by a neighbor.

Broadcasting verifiable information with regards to the Pseudo-random number generator (PRNG): We make simple modifications to the IEEE 802.11 MAC to enable the nodes in an ad hoc network discern MAC misbehavior of neighbors. All nodes use a sequence of back-off timers generated by a pseudo-random number generator (PRNG). The nodes are required to provide their respective MAC addresses as the "seed" to the PRNG. The advantage of making this modification is to make each node in the network aware (within a certain degree of ambiguity) of the pseudo-random sequence (PRS) of back-off times to be obliged by each of its neighbors during their transmissions. Since the MAC address of each node is unique and each of the nodes is aware of its one-hop neighbors, the scheme is simple and viable and does not alter the semantics of the IEEE 802.11 random back-off algorithm. The scheme does not require any hardware modifications to disseminate the PRNG and it is applicable to all CSMA networks. Note here that it is extremely important to use such a deterministic scheme in the absence of a central arbiter. Without this, a node has no knowledge with regard to the back-off times that are to be chosen by its neighbors. Furthermore, this enforcement will discourage (to a certain extent) the selfish or malicious ad hoc nodes from indulging in back-off timer violation misbehavior. A consistent deviation from the protocol by a node will alert the neighbors of the node. Dissemination of the seed of the PRNG was previously considered with the SEEDEX protocol in [20].

Every RTS packet sent by the sender S will have the *sequence-offset number* (SeqOff#) and an *attempt number* (Attempt#) included in a new field introduced in the packet. Figure 2 shows the modified packet structure for the RTS packet. This modification was done (with reference to the DCF of the IEEE 802.11 standard) in order to make sure that the nodes do not cheat in selecting the back-off values from the dictated PRS. The *sequence-offset* number (13 bits long) is to have the sender announce and therefore commit to the dictated PRS. If the sender, for its first transmission, publicizes a *sequence-offset*, it will have to increment the offset by one for the immediately succeeding transmission. An attempt to cheat in this regard will be perceived by the monitoring neighbors since they can confirm the correct sequence of expected times. The sender uses *attempt number* (3 bits long) for handling packet retransmissions. It is set to one after every successful transmission by the sender, and is incremented by one after every unsuccessful attempt. In order to verify that the sender does not cheat on this *attempt number*, a message digest (MD) of the corresponding DATA packet (to be sent) is computed using a well-known

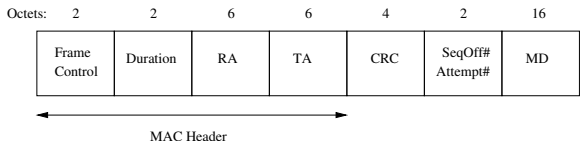


Figure 2. Modified packet structure for RTS. RA and TA are the addresses of the recipient and transmitter of RTS packet respectively. MD is the message digest of the next DATA packet.

hash function MD5 [18] and is attached as a new field in the RTS packet. The MD5 algorithm takes as input a message of arbitrary length and produces as output a 128-bit fingerprint/ message digest of the input. If a malicious node cheats on the *attempt number*, it can be discerned by verifying the MD of the DATA packet. During retransmissions, if the receiver notices a MD (for a particular DATA packet) match for multiple retransmissions, and the attempt number does not increase with the successively received transmissions, the sender is deemed to be misbehaving.

Using a statistical test to cope with scenarios of uncertainty: The monitors observe, at regular intervals, the back-off times used by their neighbors and maintain a short history of these times. For example, in Figure 1, R can verify to see if its neighbor (S) has backed off for a reasonable amount of time by comparing the relevant deterministic back-off value from the PRS to the estimated back-off value, which is computed online using the *system state*.

The monitoring node may have its own traffic to send. At these times, it is a given that the node's one hop neighbors are "not allowed" to decrement their back-off timers. Any violation of this is easily detectable.

There are times however, where the monitoring node cannot determine if the node being monitored should or should not decrement its back-off counter. Sometimes, if R senses the channel to be busy (idle) and S does not (due to interference from a third party node such as T or V), or vice-versa, R cannot know if S has to freeze (decrement) its back-off timer. During such uncertain periods, where R is not certain with regard to the state that is experienced by S, R uses its own *system state* to statistically estimate if S would be decrementing/freezing its counters. However, with such statistical estimates, it would be incorrect to diagnose a neighboring node as malicious just by making a single observation. So, a short history of the monitored back-off values is necessary to hypothesize if or not a node is misbehaving. Let 'x' be a sequential population of the known/dictated PRS of the back-off times for the node being monitored (i.e., S) and let 'y' be the sequential population of the estimated back-off times based on the obser-

vations made by R. We formulate a hypothesis test using which R attempts to determine if S is malicious:

Null Hypothesis (H0): S is well-behaved.
Alternate Hypothesis (H1): S is malicious.

The monitoring nodes can set the extent of the difference that is permissible between x and y after assessing the *system state*. A test statistic is essential to compare and see if or not the two populations (x and y) are identical. For hypothesis testing of this type, t-tests are fairly popular [19]. However, the t-tests assume that the chosen samples are from a Gaussian distribution. If the chosen sample distributions do not closely resemble the Gaussian distribution, statistical studies suggest the use of a non-parametric test called the Wilcoxon rank sum test to examine the significance of the difference between two samples. This test does not assume any prior distribution of the sample sets. An assumption of the rank sum test is that the individual sampling units are independent; this is valid in our case because each of the pseudo-random sequential back-off times is a randomly generated number. We explain the rank sum test in brief; a complete description may be found in [19].

The Wilcoxon rank sum test: The first step here is to rank all the data from both populations. Thus, the smallest value is assigned a rank 1; the second smallest is assigned a rank 2, and so on. If values are tied, they are first arbitrarily ordered in rank and are then assigned a new rank that is equal to the average of their previously assigned ranks. The ranks for each of the groups are added together (hence the term rank sum test). The rank sums are then compared by means of available tables [19]. Depending on the proximity of the rank sums, the tables yield a *significance probability* 'p' that quantifies the chance that the populations generating the two samples, x and y, are identical. If p is small, it suggests that it is unlikely that the null hypothesis is true.

5 Results

In this section, we present simulation results that quantify the performance of our proposed framework. We have focused on the case of a single malicious node to simplify the interpretations and understanding of the results[7].

Simulation Model: Simulations are performed using the event driven network simulator *ns2* (version 2.26) [15]. We extend the simulator with modifications at the MAC layer (as explained in Section 4) needed for our proposed framework. To take into account long term fading effects present in real channels [5], we have used the shadowing channel model that is represented by the following equation:

$$[\frac{P_r(d)}{P_r(d_0)}]_{dB} = -10 * \beta * \log \frac{d}{d_0} + X_{dB}$$

where d is the distance between the sender and receiver, d_0 is a reference distance, $P_r(d)$ is the mean received power at distance d, $P_r(d_0)$ is the mean received power at distance d_0, β is the path loss exponent, and X_{dB} is a Gaussian random variable with zero mean and σ_{dB} standard deviation. For free space propagation, we set $\beta = 2$ and $\sigma_{dB} = 1$.

We have done experiments with (a) a CBR traffic model (with varying data rates); a node sends a CBR stream to an arbitrarily chosen neighbor for a preset period and (b) a Poisson traffic generation model (with varying rates); each packet generated by a node is destined for an arbitrarily chosen neighbor. The results from both the cases were found to be almost identical when the traffic intensities were identical. We expect that our methods will work well even with more realistic traffic generators. Table 1 summarizes the parameters used in our simulations. ***Note that unlike with our analytical formulations, the traffic load and the contention levels experienced by different nodes can be quite different over small time-scales with this setup.*** The simulation results demonstrate that the previously made approximations do not significantly affect the performance of the approach (low false positives).

For our first set of experiments, the nodes are placed in a grid topology with 7 rows and 8 columns with 30 source-destination pairs where each source randomly chooses any of its one-hop neighbors as the destination. Results are averaged over 20 simulation runs for the computation of the probabilities derived in Section 3 and over 10,000 runs for discerning the misbehavior of a malicious node. For our second set of experiments, we consider random placements of nodes. The parameter settings are identical to those in the previous experiments except that the number of nodes is increased to 112 to ensure that the network has a high probability of being strongly connected. To model mobility, we use the random waypoint model in a 3000 m x 3000 m rectangular field with 112 nodes; the speed is uniformly distributed between 0-20m/s. We choose a neighbor of the malicious node to monitor its activity. If this neighbor moves out of range, another neighbor is randomly chosen. Simulations are run for 300 seconds.

To model various levels of misbehavior, we use the parameter the "Percentage of Misbehavior" (PM) as defined in [13]. By saying that a malicious node has an associated percentage of misbehavior of m%, we mean that it transmits a packet after counting down to (100-m)% of the dictated back-off value generated as per its PRS. Note that larger values of PM indicate greater misbehavior.

Simulation Measurements: In our experiments, sender (S) and receiver/monitor (R) are one-hop neighbors placed

[7]Note however, that our scheme is capable of detecting multiple malicious nodes (for small numbers of such nodes).

Table 1. Parameters used in simulations

Simulator	NS2 (version 2.26)
Topology types	Grid, Random
Total number of nodes	56 (Grid topology)
	112 (Random topology)
Topology Area	3000m X 3000m
Dist. Between one-hop neighbors(Grid)	240m
Transmission range	250m
Sensing/Interference range	550m
Mobility	Random waypoint model
Range of speed	0-20 m/s
Pause times	0,50,100,200,300 seconds
Traffic Model	Poisson, CBR
Queue length	50
Packet size	512 bytes
Simulation time	300s
Physical, MAC Layers	IEEE 802.11 specs.
Routing protocol	AODV
Transport protocol	UDP

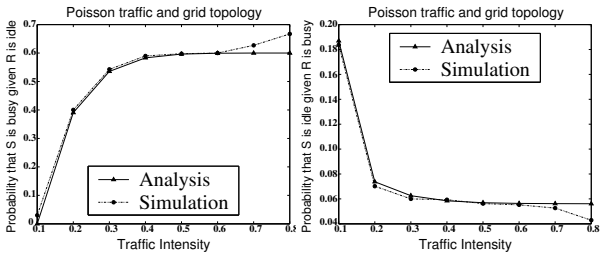
in the center of the grid so that the computations take into consideration the interference effects from their two-hop neighbors. In the first set of experiments, we have deterministically set n=5, k=5, since they are fixed in the grid topology[8]. In the second set of experiments, the monitor estimates n and k (as discussed earlier) for each run.

Analytical computations for $P_{B/I}$ and $P_{I/B}$ are plotted versus the variation in traffic intensity in Figure 3(a) and Figure 3(b), respectively. In these experiments, all the nodes on the grid were well behaved. The total number of busy and idle slots sensed by S and R were recorded for varying levels of traffic intensity. These observations were made over a sequential series of 50000 time-slots (each slot is 20μs as defined by the IEEE 802.11 MAC) for each computation instance. From these numbers, the conditional probabilities $P_{B/I}$ and $P_{I/B}$ are computed.

In Figure 3(a), with an increase in traffic intensity in the network, we see that the probability that S senses the channel to be busy increases. This is due to a higher possibility of transmissions from a node's neighbors in the network. Irrespective of the state of R, one might expect that S would sense the channel to be busy for larger proportions of the time with an increase in traffic intensity. We point out here that the probability that R senses the channel to be idle decreases at higher traffic loads. Figure 3(a) also depicts the results of our simulation experiments; we see that the analytical computations conform to the simulation results thereby demonstrating that the approximations that we make in the analysis are reasonable.

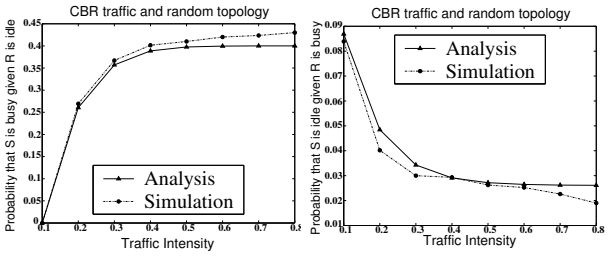
In Figure 3(b), note that as the traffic intensity increases, the probability that S senses the channel to be idle decreases even if R senses the channel to be busy. This is because, the nodes that are in S's sensing range (but beyond the sensing range of R) have a higher chance of being involved in transmissions as the intensity increases. Consequently, a lower probability is associated with the event that S senses the channel to be idle. Again, analytical results conform to observations from simulation results.

[8]Experiments were carried out for different higher values of n, k. We found that these parameters do not play a significant role in the computation of necessary probabilities.



(a) (b)

Figure 3. (a) Probability that S senses the channel to be busy when R senses it to be idle: Poisson traffic, grid topology. (b) Probability that S senses the channel to be idle when R senses it to be busy: Poisson traffic, grid topology.

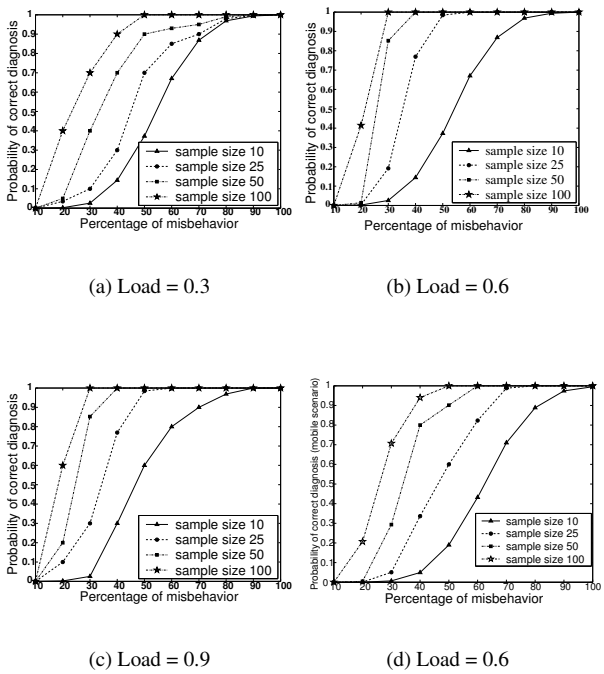


(a) (b)

Figure 4. (a) Probability that S senses the channel to be busy when R senses it to be idle: CBR traffic, random topology. (b) Probability that S senses the channel to be idle when R senses it to be busy: CBR traffic, random topology.

We plot the analytical and simulation results for these probabilities, but with CBR traffic and the random topology, in Figure 4(a) and Figure 4(b). The observations are similar to those with the grid topology. The previous discussion on the behavior of the results for the grid topology holds in this case as well. Next, we measure the following to evaluate our proposed scheme:

- *The probability of correct diagnosis of a misbehaving node*: This is the probability of discerning a truly misbehaving node using hypothesis testing. For this purpose, S was simulated to behave maliciously while all the other nodes were well behaved. While it is true that all of S's neighbors moni-



(a) Load = 0.3 (b) Load = 0.6

(c) Load = 0.9 (d) Load = 0.6

Figure 5. Probability of correct diagnosis for varying levels of node misbehavior. (a)-(c): Static grid topology. (d) With mobility.

tor its behavior, for simplicity, we consider a single monitoring node, R.

- *The probability of misdiagnosis*: This is the probability of wrongly diagnosing a well-behaved node as malicious. For this purpose, all the nodes, including S, participating in the simulation scenario were made to behave well.

Our null hypothesis states that S is well behaved. We measure the probability of rejecting this hypothesis, i.e., the probability of correctly diagnosing S as misbehaving. The monitoring node observes the behavior of the node being monitored over a certain number of sampled back-off times.

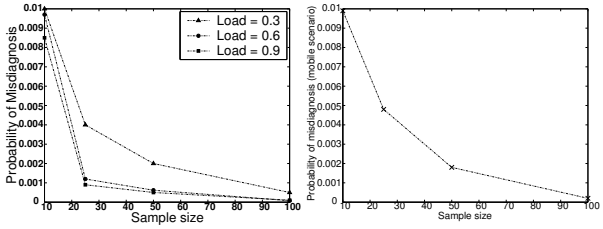
Figure 5 depicts the probability of correct diagnosis using four different sample sizes (10, 25, 50 and 100) when the grid topology is used[9]. Different loads are considered. For ease of discussion, we first focus on the case wherein the load is 0.6. As seen from the graph, it is easier to detect a malicious node if its PM is higher. Our scheme detects a node that misbehaves by reducing its computed back-off time to about 35% of the time (i.e., PM = 65 %) with a probability that is higher than 0.8, even with a small sample size of 10. On the other hand, with a larger sample size of 100 the node can correctly detect a node that misbehaves by reducing its computed back-off to approximately 75% of the dictated time (PM = 25 %) with a probability close to one. With an increase in the sample size, the accuracy improves significantly, but note that, it now takes longer time to record a bigger history. So, there is a trade-off between the "quickness" of detection and the accuracy with which a malicious node is detected. The figure also demonstrates that our approach is viable even if the load in the network were to be varied. At low loads, however, it is to be noted that it takes a longer time to collect the required samples (as an example, collecting 25 samples at a load of 0.3 takes longer than it takes to collect a sample at a load of 0.9). Thus, it takes longer to detect node misbehavior. However, we wish to point out that misbehavior is less of a concern at low loads than when the network is being heavily utilized. It is in this regime that our methods are most effective.

In Figure 5(d), the performance of our scheme in mobile scenario is shown (with a load of 0.6). Note that timer violations are effectively discovered. However, a large number of samples (approximately twice the number) are required for convergence as compared to the case wherein there was no mobility. This is a consequence of the topological changes due to mobility. The positions of the nodes no longer accurately conform to a uniform distribution. However, we wish to point out that our methods, although approximate, are very effective in achieving their objective.

Inaccurately diagnosing a node as being malicious is referred to as a false alarm or misdiagnosis. To compute the misdiagnosis probability, we simulated a scenario in which all the nodes are well behaved. Since the null hypothesis deems the node S to be well behaved, again, we measure the probability of rejecting the null hypothesis. In this case, this is the probability of misdiagnosis. Figure 6(a) depicts this probability as a function of the sample size when the grid topology is used. The maximum misdiagnosis probability was found to be a little less than 0.01 when a history of 10 sample values is maintained and considered. With an increase in sample size, the accuracy improves considerably and the misdiagnosis probability further reduces. Note that misdiagnosis is higher at lower loads; this is due to the fact that local variations are higher (longer times are needed to collect samples) in these regimes. However, the probability is still very small. The effectiveness of our methods improve with load and are most effective at high loads wherein misbehavior is of most consequence. Figure 6(b) depicts the probability of false alarms, with mobility (with a load of 0.6[10]). Note that a sample size of 50 is sufficient to maintain this probability below 0.2%.

[9]The results observed, with the random topology and with either Poisson or CBR traffic, were similar and are therefore omitted.

[10]Results with other loads suggest trends that are similar to that with the static case.



(a) (b)

Figure 6. Probability of misdiagnosis measured over varying sample sizes. (a) Static grid topology. (b) With mobility (Load = 0.6).

6 Conclusions

In this paper, we focus on the problem of detecting back-off timer violations with the IEEE 802.11 MAC protocol in ad hoc networks. We propose a framework that is based on a combination of deterministic and statistical methods that allow nodes to discern violations of back-off timers by neighboring nodes. First, the nodes are required to exchange the state of their pseudo-random number generators with their neighbors. This allows a node to know the sequence of back-off times that are to be used by each of its neighbors. However, in certain scenarios, due to interference effects, a node may not be able to accurately monitor the back-off countdown of a neighbor. In order to cope with this, we use statistical online estimates to compute the expected back-off time. Our online estimation is based on each node computing an ARMA of observations to assess the offered traffic intensity and the density of nodes in its localized neighborhood. We show by means of extensive simulation studies that our framework can provide accurate assessments of node misbehavior within short observation periods with extremely low probabilities of false alarms.

References

[1] Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. IEEE STD 802.11.

[2] G. Bianchi. Performance Analysis of the IEEE 802.11 Distributed Coordination Function. *IEEE Journal on Selected Areas in Communications*, 18(3), 2000.

[3] G. Bianchi and I. Tinnirello. Kalman Filter Estimation of the Number of Competing Terminals in an IEEE 802.11 network. In *Proceedings of IEEE INFOCOM*, 2003.

[4] J. Broch, D. Maltz, D. Johnson, Y. Hu, and J. Jetcheva. A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocol. In *Proceedings of ACM/IEEE MOBICOM*, 1998.

[5] K. Fall and K. Varadhan. ns notes and documentation. Technical report, UC Berkeley, LBL, USC/ISI, Xerox PARC, 2002.

[6] V. Gupta, S. Krishnamurthy, and M. Faloutsos. Denial of Service Attacks at the MAC Layer in Wireless Ad Hoc Networks. In *Proceedings of IEEE MILCOM*, 2002.

[7] Y. Hu, A. Perrig, and D. Johnson. Ariadne: A Secure On-demand Routing Protocol for Ad Hoc Networks. In *Proceedings of ACM/IEEE MOBICOM*, 2002.

[8] Y. Hu, A. Perrig, and D. Johnson. Packet Leashes: A Defence against Wormhole Attacks in Wireless Networks. In *Proceedings of IEEE INFOCOM*, 2002.

[9] L. Kleinrock. *Queueing Systems: Volume I: Theory*. John Wiley and Sons, 1975.

[10] J. Kong and X. Hong. ANODR: ANonymous On Demand Routing with Untraceable Routes for Mobile Ad-hoc Networks. In *Proceedings of ACM MOBIHOC*, 2003.

[11] J. Konorski. Protection of Fairness for Multimedia Traffic Streams in a Non-cooperative Wireless LAN Setting. *PROMS*, 2213, 2001.

[12] J. Konorski. Multiple Access in Ad-Hoc Wireless LANs with Noncooperative Stations. *NETWORKING*, 2345, 2002.

[13] P. Kyasanur and N. Vaidya. Detection and Handling of MAC Layer Misbehavior in Wireless Networks. In *Proceedings of Dependable Systems and Networks*, 2003.

[14] S. Marti, T. Giuli, K. Lai, and M. Baker. Mitigating Routing Misbehavior in Mobile Ad Hoc Networks. In *Proceedings of ACM/IEEE MOBICOM*, 2000.

[15] S. McCanne and S. Floyd. Ns-2 simulator. http://www.isi.edu/nsnam/ns/.

[16] P. Michiardi and R. Molva. Game theoretic analysis of security in mobile ad hoc networks. Technical report, Institut Eurecom, 2002.

[17] M. Raya, J. Hubaux, and I. Aad. DOMINO: A System to Detect Greedy Behavior in IEEE 802.11 Hotspots. In *Proceedings of MOBISYS*, 2004.

[18] R. Rivest. The MD5 Message-Digest Algorithm. RFC 1321.

[19] V. Rohatgi and A. Saleh. *An Introduction to Probability and Statistics*.

[20] R. Rozovsky and P. Kumar. SEEDEX: A MAC protocol for ad hoc networks. In *Proceedings of ACM MOBIHOC*, 2001.

[21] K. Sanzgiri, B. Dahill, B. Levine, C. Shields, and E. Royer. A Secure Routing Protocol for Ad Hoc Networks. In *Proceedings of IEEE ICNP*, 2002.

[22] S. Sundaramurthy and E. Royer. The AD-MIX protocol for Encouraging Participation in Mobile Ad Hoc Networks. In *Proceedings of IEEE ICNP*, 2003.

[23] H. Yang, H. Luo, F. Ye, S. Lu, and L. Zhang. Security in Mobile Ad Hoc Networks: Challenges and Solutions. *IEEE Wireless Communications*, 2004.

[24] S. Yi and R. Kravets. MOCA: Mobile Certificate Authorhity for wireless ad hoc networks. In *Proceedings of 2nd Annual PKI Research Workshop*, 2003.

[25] M. Zapata and N. Asokan. Securing Ad Hoc Routing Protocols. In *Proceedings of ACM WISE*, 2002.

[26] L. Zhou and Z. Haas. Securing Ad Hoc Networks. *IEEE Network Magazine*, 13(6), 1999.