

TECHNICAL RESEARCH REPORT

Detection and Prevention of MAC Layer Misbehavior for Ad Hoc Networks

by Alvaro A. Cardenas, Svetlana Radosavac, John S. Baras

**SEIL TR 2004-4
(ISR TR 2004-30)**



ISR develops, applies and teaches advanced methodologies of design and analysis to solve complex, hierarchical, heterogeneous and dynamic problems of engineering technology and systems for industry and government.

ISR is a permanent institute of the University of Maryland, within the Glenn L. Martin Institute of Technology/A. James Clark School of Engineering. It is a National Science Foundation Engineering Research Center.

Web site <http://www.isr.umd.edu>

Alvaro A. Cárdenas, Svetlana Radosavac and John S. Baras

Electrical and Computer Engineering Department

and the Institute for Systems Research

University of Maryland College Park

acardena, ,svetlana,baras@isr.umd.edu

Revised

November 19, 2004

Abstract

Selfish behavior at the MAC layer can have devastating side effects on the performance of wireless networks, similar to the effects of DoS attacks. In this paper we focus on the prevention and detection of the manipulation of the backoff mechanism by selfish nodes in 802.11. We first propose an algorithm to ensure honest backoffs when at least one, either the receiver or the sender is honest. Then we discuss detection algorithms to deal with the problem of colluding selfish nodes. Although we have focused on the MAC layer of 802.11, our approach is general and can serve as a guideline for the design of any probabilistic distributed MAC protocol.

1 Introduction

The communication protocols of different layers of an ad hoc network such as the medium access control (MAC) protocol, the routing protocol and the transport protocol, were designed under the assumption that all nodes would obey the given specifications. However when these protocols are implemented in an untrusted environment, nodes can deviate from the protocol specification in order to obtain a given goal, at the expense of honest participants. A selfish user can disobey the rules to access the wireless channel in order to obtain a higher throughput than the other nodes. A selfish user can also change the congestion avoidance parameters of TCP in order to obtain unfair advantage over the rest of the nodes in the network. In limited power devices, certain nodes might refuse to forward packets in behalf of other sources in order to save battery power. In all these cases, the misbehaving nodes will degrade the performance of the network from the point of view of the honest participants. To fully address these problems, a layered security mechanism should be deployed in order to enforce cooperation or to penalize misbehaving nodes. In this paper we focus on the prevention and detection of unfairness and collision of packets, caused by selfish users at the MAC layer in ad hoc networks.

The MAC layer in a communication network manages a multiaccess link (e.g. a wireless link) so that frames can be sent by each node without constant interference from other nodes. A fairly used MAC protocol for wireless networks is the IEEE

802.11 MAC protocol, which uses a distributed contention resolution mechanism for sharing the wireless channel. Its design tries to ensure a relatively fair access to the medium for all participants of the protocol. In order to avoid collisions, the nodes follow a binary exponential backoff scheme that favors the last winner amongst the contending nodes. One problem with the 802.11's MAC protocol is that even when all contending nodes are well behaved, this mechanism can lead to short time unfairness under the capture effect: nodes that are heavily loaded tend to capture the channel by continuously transmitting data making lightly loaded neighbors to backoff continuously. Very similar effects are obtained when one of the contending nodes is selfish.

MAC layer misbehavior is possible in network access cards that run the MAC protocol in software rather than hardware or firmware, allowing a selfish user or attacker to easily change MAC layer parameters. Even network interface cards implementing most MAC layer functions in hardware and firmware usually provide an expanded set of functionalities which can be exploited to circumvent the limitations imposed by the firmware [2]. In the worst case scenario a vendor might create NIC cards violating the MAC protocol to create an improved performance of its products.

A selfish node in the MAC layer will try to maximize its own throughput and therefore will keep the channel busy. As a side effect of this behavior, regular nodes cannot use the channel for transmission, which leads to a denial of service (DoS) attack [9]. A selfish user can implement a whole range of strategies to maximize its access to the medium. The most likely strategy that a selfish user will employ is to use different schemes for manipulating the rules of the MAC layer. In 802.11, the attacker can manipulate the size of the Network Allocation Vector (NAV) and assign large idle time periods to its neighbors, it can decrease the size of Interframe Spaces (both SIFS and DIFS), it can select small backoff values, it can deauthenticate neighboring nodes etc. A successful detection scheme should take into account all possible cheating options in the MAC layer and detect both: users that employ only one scheme and users that employ a combination of several schemes (e.g. first choosing small backoff values, then assigning large NAV values to its neighbors etc).

One of the most challenging detection tasks is that of detecting backoff manipulation [13, 2]. Due to the randomness introduced in the choice of the backoff, it is difficult to detect when a node has chosen small backoff values by chance or not. In this work we focus on prevention and detection of the manipulation of the backoff mechanism of 802.11's MAC protocol, although our approach can be extended to any probabilistic distributed MAC protocol.

The organization of this paper is the following. The next section summarizes related work. Section 3 presents an introduction to the MAC protocol 802.11 DCF.

In section 4 we present an algorithm that prevents cheating in the backoff stage of 802.11 DCF for non-colluding nodes. In Section 5 we present algorithms for detecting misbehavior of colluding nodes. In the last section we discuss future research directions and conclude the paper.

2 Related Work

Selfish misbehavior at the MAC layer has been addressed mostly from a game theoretic perspective considering all nodes are selfish. The goal in a game theoretic setting is to design distributed protocols that guarantee for each node, the existence, uniqueness and convergence to a Nash equilibrium with an acceptable throughput. As we have previously pointed out, if users try to maximize their throughput, every node will attempt to transmit continuously in such way that users will deny access to any other node and the network would collapse. This network collapse due to aggressive selfish behavior is a Nash equilibrium. In order to obtain a different Nash equilibrium, each node needs to be assigned a cost for each time it accesses the channel. For example in [12, 1], they consider the case of selfish users in Aloha that attempt to maximize their throughput and minimize the cost for accessing the channel (e.g. energy consumption). Another game theoretic scheme for CSMA/CA schemes is presented in [6]. It shows how a Nash equilibrium is achieved among selfish users when the cost for accessing the channel repeatedly is being jammed by another node. A node jams anonymously any other node that achieves higher throughput than the average of everyone else (assuming nodes always have data to transmit, the throughput of every node should be fair). They assume all nodes are within wireless range in order to avoid the hidden terminal problem, so this scheme is mostly intended for wireless LANs.

Since game theoretic protocols assume all nodes are selfish (the worst case scenario), the throughput achieved in these protocols is substantially less than in protocols where the honest majority cooperates. Under the assumption of an honest majority, detection of misbehaving nodes becomes the primary goal in dealing with misbehavior.

Several possible schemes of node misbehavior in 802.11 for achieving a higher throughput are presented in [13]. The detection of such misbehavior is achieved through a system called DOMINO. However, their detection scheme for backoff manipulation is a suboptimal detection technique for every strategy of the greedy user. In section 5 we propose new detection schemes for the backoff manipulation that we believe will improve the performance of systems such as DOMINO.

Kyasanur and Vaidya [11] propose a modification to 802.11 for facilitating the detection of misbehaving nodes. In their scheme, the receiver (a trusted host -e.g.

base station-) assigns the backoff value to be used by the sender, so the former can detect any misbehavior of the latter and penalize it by increasing the backoff values for the next transmission. The protocol consists of Detection, Penalty and Diagnosis Schemes. The sender is considered to be deviating from the protocol if the observed number of idle slots, B_{act} , is smaller than a specified fraction α of the assigned backoff B_{exp} . For a detected node, a penalty for the next assigned backoff is selected given a measure of the deviation $D = \max(\alpha B_{exp} - B_{act}, 0)$. If the sender deviates repeatedly, i.e. if the sum of misbehavior in a sliding window is bigger than some threshold, then the sender is labeled as misbehaving and the receiver takes drastic measures, e.g. drop all packets by the sender.

The problem of applying this protocol for ad hoc networks is that the receiver might not be trusted. In section 4 we extend the idea of [11] by presenting an algorithm that ensures a honest backoff selection among the sender and a receiver as long as one of the participants does not misbehave.

All the schemes presented above as well as the ones we propose, require the proper use of MAC layer authentication schemes, providing uniquely verifiable identities in order to prevent impersonation and Sybil attacks [8].

We also assume that there is a reputation management system similar to CONFIDANT [5, 4], where nodes can monitor and distribute reputation values about other nodes behavior at the MAC layer (CONFIDANT however focuses in reputation at the routing layer). The design of a robust MAC layer reputation system and response is essential and one of our main topics of future work.

3 IEEE 802.11 DCF

The distributed coordinating function (DCF) of 802.11 specifies the use of CSMA/CA to reduce packet collisions in the network. A node with a packet to transmit picks a random backoff value b chosen uniformly from the set $\{0, 1, \dots, CW - 1\}$ (CW is the contention window size), and transmits after waiting for b idle slots. Nodes exchange request to send (RTS) and clear to send (CTS) packets to reserve the channel before transmission. Both the RTS and the CTS contain the proposed duration of data transmission: the duration field indicates the amount of time (in microseconds) after the end of the present frame that the channel will be utilized to complete the successful transmission of the data or management frame. Other hosts which overhear either the RTS or the CTS are required to adjust their network allocation vector (NAV), which indicates for how long should the node defer transmissions on the channel, including the SIFS interval and the acknowledgment frame following the transmitted data frame. If a transmission is unsuccessful (by the lack of CTS or the ACK for the data sent), the CW value is doubled. If the

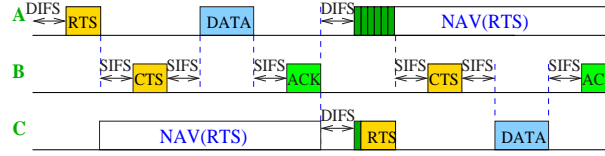


Figure 1: Nodes A and C contend for accessing node B. The first time A reserves the channel, and in the second time C accesses the channel.

transmission is successful the host resets its CW to a minimum value CW_{min} .

Fig. 1 shows an example of contending nodes using the protocol.

Typical parameter values for the MAC protocol depend on the physical layer that 802.11 uses. For example table 1 shows the parameters used when the physical layer is using direct sequence spread spectrum (DSSS).

DIFS	$50\mu s$
SIFS	$10\mu s$
SlotTime	$20\mu s$
ACK	$112\text{bits}+\text{PHY_header}=203\mu s$
RTS	$160\text{bits}+\text{PHY_header}=207\mu s$
CTS	$112\text{bits}+\text{PHY_header}=203\mu s$
DATA	$\text{MAC_header (30b)}+\text{DATA(0-2312b)}+\text{FCS(4b)}$
Timeouts	$300-350\mu s$
CW_{min}	32 time slots
CW_{max}	1024 time slots

Table 1: Parameters for DSSS

4 ERA-802.11: Ensuring Randomness for 802.11

As we have discussed before [11] requires the receiver to be trusted. This assumption is well suited for infrastructure-based wireless networks, where the base station can be trusted. However, in the case of ad hoc networks the receiver can misbehave by selectively assigning the backoff values to different senders. Depending on the concrete situation, a receiver may benefit by assigning small backoff values to a particular sender (when data from that particular sender needs to be received) or by assigning large backoff values to different neighbors (when it wants to degrade overall performance of neighbors and improve its own throughput). Furthermore, existence of multiple sender-receiver pairs in the interference range of each other

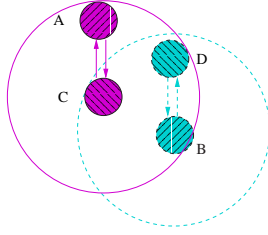


Figure 2: Node C transmits to A and node B wants to transmit to D. After hearing the backoff assigned by A to C, node D assigns a backoff to node B such that it collides with C.

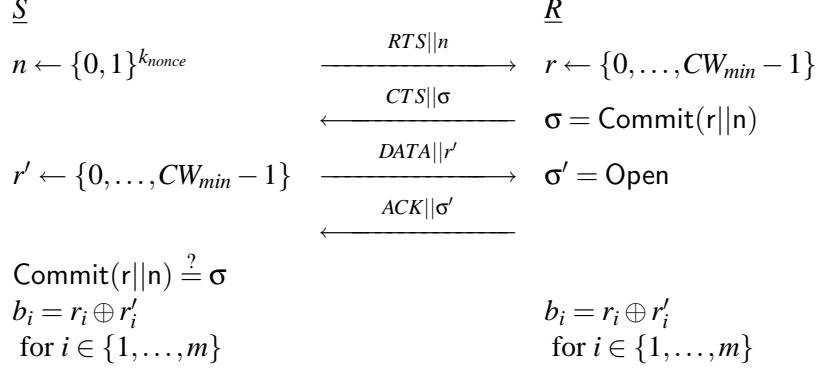
creates additional security issues. More specifically, a malicious receiver D in Fig. 2 can overhear the backoff value assigned to node A by node C and unilaterally select a backoff for node B in order to create a collision with C .

In this section we propose an extension to the 802.11 CSMA/CA protocol that ensures a uniformly distributed random backoff, when at least one of the parties is honest. The basic idea follows the protocol for flipping coins over the telephone by Blum [3]. The goal is that the sender and the receiver agree through a public discussion on a random value. The main property of the protocol is that an honest party will always be sure that the agreed value is truly random. For an honest sender this means that he can expect a fair treatment in order to access the channel. On the other hand an honest receiver can monitor the behavior of the sender and report a misbehaving node to the reputation management system.

To detect sender deviation from the agreed backoff, the detection algorithm given in [11] can be used.

It is also mentioned in [11] how to handle the detection during packet retransmissions, i.e. when the sender collides and has to choose by itself another backoff value from the set $\{0, 1, \dots, (CW_{min} + 1)2^{i-1} - 1\}$ (where i is the number of retransmissions). Recall that CW keeps increasing until it reaches CW_{max} . Another way to deal with packet retransmissions is to change our protocol such that instead of agreeing on a given backoff, the sender and the receiver would agree on a seed for a pseudorandom generator. In this way a receiver would know the future backoffs, even in the case of retransmissions. Another advantage for the sender and the receiver to agree on a seed value is that they do not need to execute our protocol for every reservation of the channel.

The protocol we propose can be embedded in 802.11 and used every time a new reservation of the channel takes place. The messages are appended (denoted by a double bar $||$) to the normal message exchange of 802.11:



We now explain the protocol step by step.

1. In the first step the sender S selects a nonce: a number n selected uniformly from the set $\{0, 1, \dots, 2^{k_{nonce}}\}$, denoted as $n \leftarrow \{0, 1\}^{k_{nonce}}$. k_{nonce} is a security parameter indicating the level of difficulty of guessing n . For example $k_{nonce} = 64$. This step is optional and is done in order to prevent an offline attack on the commitment scheme.
2. In the second step the receiver R selects a random backoff r from the set $\{0, 1, \dots, CW_{min} - 1\}$ and commits to it. In binary notation r is a random bit string of length m ($r = r_1 r_2 \dots r_m$), where $m = \log_2 CW_{min}$ (note that the contention window size CW is always a power of two). The commitment scheme Commit is such that the following two properties are satisfied (at least before the time-out for channel reservation: $300\mu s - 350\mu s$):
 - Binding:** After sending $\text{Commit}(r||n)$, the receiver cannot open the commitment to a different value $\tilde{r} \neq r$ (except with negligible probability). This protects against a dishonest R that might try to change the committed value depending on the r' received by S .
 - Hiding:** Given $\text{Commit}(r||n)$, S cannot extract any information about r that will enable it to distinguish r from any other bit string of length m (except with negligible probability). This protects against a dishonest S that will try to tailor r' based on its guess of r .
3. After receiving the Commitment σ , S selects a random value $r' = r'_1 r'_2 \dots r'_m$ from $\{0, 1, \dots, CW_{min} - 1\}$.

4. Finally R opens its commitment to S . Opening a commitment is an operation that reveals the committed value r and some additional information to S . This enables the other party to verify that the revealed and committed values are the same. If the value opened by the R is correct, both sender and receiver compute the backoff $b = b_1 b_2 \cdots b_m$ as the xor of the bits: $b_i = r_i \oplus r'_i$. Otherwise, the sender can report misbehavior of the node to the reputation management system.

Several commitment schemes are known under very different computational assumptions. Very efficient commitment schemes in terms of computation and communication, can be implemented under the random oracle model. In this setting it is a standard practice to assume that hash functions H such as SHA-1 or MD5 are random oracles. Under this assumption it is easy to confirm that the following commitment scheme satisfies the binding (by assuming H is collision resistant) and hiding properties (by assuming H is a random oracle):

$$\begin{array}{l} \underline{\text{Commit}(r||n)} \\ i \leftarrow \{0, 1\}^k \\ \text{Output} = H(i||r||n) \\ \\ \underline{\text{Open}} \\ \text{Output} = (i, r) \end{array}$$

where k is a security parameter (e.g. $k = 64$, since it is not considered feasible to search for 2^{64} messages given the current state of art). To open the commitment, R has to send both r and i so that S can check validity of the commitment.

We now consider 802.11 with Direct Sequence Spread Spectrum (DSSS) physical layer. In DSSS mode the minimum contention window size is 32 time slots, therefore $m = \log_2 CW_{min} = 5$, that is, r' and r are only 5 bits long which is an insignificant quantity to be appended to a *DATA* frame. The acknowledgement frame is appended $k + m = 69$ bits.

If we use SHA-1 to implement the hash function of the commitment then we obtain a message digest of 160 bits. The *CTS* frame is doubled in size if the full message digest is used. If doubling the size of a *CTS* frame is a concern, the output of SHA-1 can always be truncated (for example to 80 bits). The security reduction of the message digest has to be evaluated under the birthday paradox: if the message digest has h bits, then it would take only about $2^{h/2}$ messages (out of $2^{k+m+k_{nonce}}$), chosen at random, before one would find two (inputs) with the same value (message digest). Considering the normal timeout between frames to be $300\mu s$, we can safely assume 2^{40} computations cannot be done in this time. Finally the nonce parameter should discourage offline attacks, e.g. $k_{nonce} = 64$.

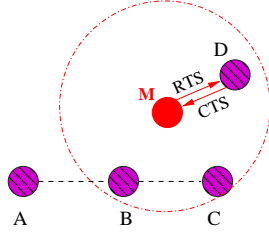
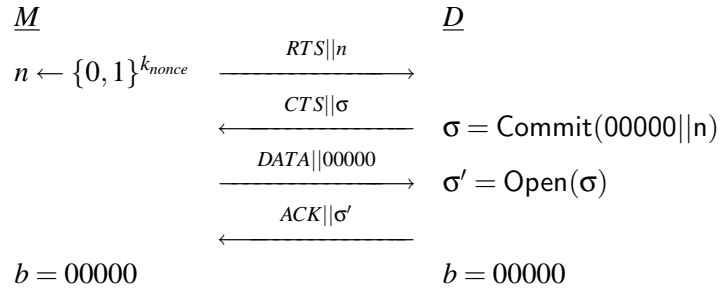


Figure 3: Nodes M and D collude and interfere in the communication path of nodes B and C

Note that in the case the sender and receiver agree on a seed for a pseudorandom number generator the parameters k and k_{nonce} can be avoided if the seed is long enough.

5 Detection System

The algorithm presented in Section 4 is not resistant to colluding nodes. When sender and receiver collude by selecting their backoff a priori, they can deny access to the network to neighboring nodes. For example, consider Fig. 3 and assume C is within wireless range of nodes D and M (it is a reasonable assumption that in wireless networks there will always be nodes that are neighbors of both colluding nodes: D and M). Without loss of generality, assume C monitors the access times of node M . Note that C can compute the exact backoff value of its neighboring nodes by listening to the exchanged values n, σ, r', σ' (between M and D) and then computing the backoff $b_i = r_i \oplus r'_i$. If the sender deviates from this backoff then node C can detect a misbehaving sender in the same way a honest D would detect a misbehavior of M . However, if nodes D and M collude, they can select their numbers a priori. For example they can collude to present a valid message agreement on the backoff zero to node C by selecting the following values:



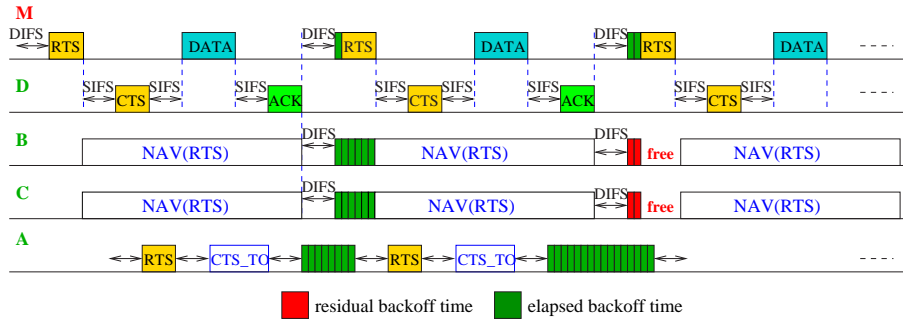


Figure 4: Nodes M and D collude and select a very small backoff, thereby denying access to node A by causing CTS timeouts.

In Fig. 4 we show how the sequence of small backoffs $0, 1, 2, \dots$ from node M causes the CTS timer of node A to time out. Node A will therefore repeatedly backoff exponentially, decreasing its chances for accessing the network. This setting was simulated in the network simulator Opnet such that the colluding nodes transmit with no backoff in a time period. Fig. 5 shows how node A is denied network access while colluding nodes communicate.

Having motivated the need to detect colluding MAC layer misbehavior in ad hoc networks, we now focus on a misbehavior detection mechanism. More specifically, we are interested in designing algorithms for detection of random backoff violations. Although our emphasis is on ad hoc networks, the same algorithms can be applied for monitoring greedy behavior at WLAN access points using the original 802.11 protocol (802.11 without modifications introduced in the previous section).

5.1 Test for backoff manipulation

We now consider detection strategies in the presence of an intelligent misbehaving node, i.e. a node that is aware of the existence of monitoring neighboring nodes and will adapt its behavior in order to avoid detection. In general, we assume that the colluding nodes are:

1. *knowledgeable*: they know everything a monitoring node knows about the detection scheme.
2. *intelligent*: they can make inferences about the situation in the same way as the monitoring nodes can.

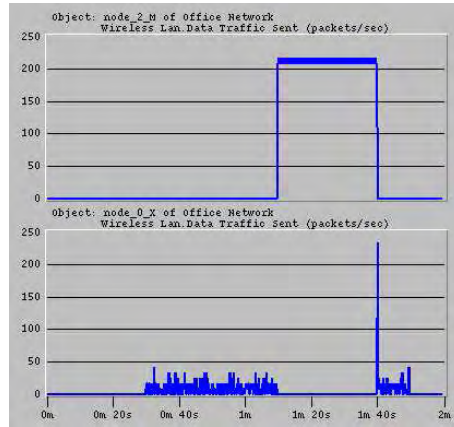


Figure 5: Simulation of traffic sent by node M (top figure) versus traffic sent by node B (top figure). When D and M collude A is denied access to the network.

The goal of this class of nodes is to avoid the probability of misbehavior detection (P_D) while maximizing their own throughput.

However, it is difficult to come up with a universal strategy that the misbehaving nodes will use for achieving this goal. A naive intrusion detection system (IDS) may assume that the misbehaving nodes will select all their backoff values to be very small. Therefore a model the IDS can assume for the attack is that the misbehaving nodes select their backoff uniformly from the set $\{0, 1, \dots, CW_{min}/4\}$. Given this model the IDS would raise an alarm when any of the monitored nodes do not backoff an amount larger than $CW_{min}/4$, after K observations (where K is chosen given an acceptable false alarm rate). However, an intelligent misbehaving node will easily defeat our detection mechanism by selecting a backoff of zero $K - 1$ times and selecting a value above $CW_{min}/4$ as the K th backoff.

As we mentioned before, another way we can use this protocol is to commit to a seed for a pseudorandom number generator.

5.1.1 Tests for change in the mean

The first intuitive assumption to make about the strategy of the colluding nodes is that it will let one of them access the channel in a way that the mean access should decrease from the minimal mean backoff $B_{min} := (CW_{min} - 1)/2$, i.e. on average the selfish node will attempt to access the channel more frequently than any other contending node. In order to also penalize nodes that do not double their contention window every time they collide, we could test for a decrease from

a nominal backoff B_{nom} (where $B_{nom} \geq B_{min}$) representing our long term average backoff. Note that each monitoring node has to estimate B_{nom} online. The selection of testing either a decrease in B_{min} or B_{nom} will depend on the risk assessment of the threat provided by the misbehaving nodes. Without loss of generality and in order to be conservative with our detection mechanism we select to test for deviations from B_{min} .

Let X_i denote the i^{th} backoff time for a given monitored node. After measuring n backoff times for node M , we end up with the sequence X_1, \dots, X_n . We assume the IDS makes a decision after n observations. Using this sequence of data, in DOMINO [13] a detection mechanism is proposed for testing a deviation from the reference backoff. The algorithm first computes an average $X_{ac} = \sum_{i=1}^n X_i/n$, of the observations taken over a given unit of time (e.g. 10s). After that, the averaged value is compared to the reference backoff :

$$X_{ac} < \gamma B_{min}$$

the parameter γ ($0 < \gamma < 1$) controls the rate for false alarms. An optional parameter $Cheat_{count}$ will only flag a false alarm after the statistic X_{ac} has exceeded γB_{min} , K times (i.e. if $Cheat_{count} = K$). A forgetting factor is considered for the cheat count, so if a node behaves normally for a given observation period, it is partially forgiven: $Cheat_{count} = Cheat_{count} - 1$ (As long as $Cheat_{count}$ remains greater than zero). The authors tested the detection scheme against a misbehaving node whose strategy was to select backoff values uniformly from the set $\{0, 1, \dots, \lfloor CW_{min} \times \delta \rfloor\}$, where δ ($0 \leq \delta \leq 1$) represents the amount of misbehavior ($\delta = 0$ meaning that the station transmits without backoff and $\delta = 1$ meaning no misbehavior). The results show that when the misbehaving node increases its throughput three times more the normal value, then the detection mechanism will always catch him while maintaining a false alarm rate below 0.1.

In an earlier version of this paper [7] we gave an adversarial strategy such that the misbehaving node does not follow the normal backoff window and is not detected by DOMINO. However the problem is that our previous strategy does not increase the throughput from the misbehaving node significantly. However, a misbehaving node in 802.11 (or a pair of colluding nodes in ERA-802.11) can still avoid detection and achieve access to the channel more than half of the contending trials by selecting the following backoff scheme: Select a zero backoff for the first $(K - 1)n$ times (this exploits the fact that an alarm is never raised until the mean statistic has exceeded γB_{min} , K times). After that the optimal strategy to avoid the detection mechanism is to alternate between acting normally $(K - 1)n$ times (e.g. select the backoff γB_{min} during that period) and acting selfishly (select a backoff of zero for the next $(K - 1)n$ times).

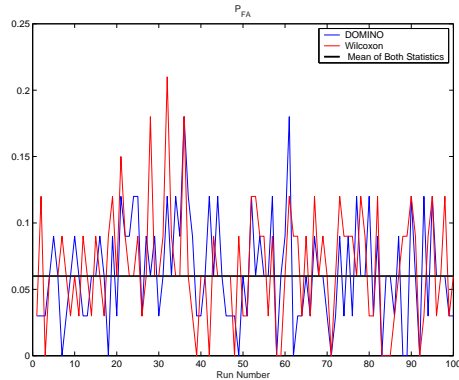


Figure 6: Several runs of DOMINO and Wilcoxon tests for an honest node in saturation condition give a mean false alarm rate of 0.06 for both tests

This strategy will ensure that an alarm is never raised for the misbehaving nodes, while still providing access to the channel more than half of the times regardless of the number of contending nodes. Although this problem can be alleviated by instructing the monitoring node to start collecting statistics for every node at a random time (so the adversary cannot determine the start of each monitoring period), we believe the detection strategy is still heuristic. There are several nonparametric tests with provably optimal performance guarantees and analytical expressions for the probabilities of false alarms and probability of detection. Several nonparametric tests measure how “shifted” in the X axis is the alternative distribution (the attack strategy) from the normal distribution (which is assumed to be symmetric at zero). If we define the independent and identically distributed (i.i.d.) random process: Y_1, \dots, Y_n by: $Y_i = B_{min} - X_i$ we can use a Sign Test (a test for the median) or a Wilcoxon Test (a test for when the alternate distribution is not symmetric at zero) [10].

To compare the resilience of the detection statistics against the strategy of a misbehaving node shifting between one period of zero backoffs and another period behaving normally (selecting a backoff of γB_{min} in order to appear normal) we simulated a scenario with only one transmitter under saturation condition, and a receiver. We first simulated an honest user in order to obtain the false alarm rate. For DOMINO we selected the same parameters as in [13]: $\gamma = 0.9$ and $K = 3$. We then tuned the threshold of the Wilcoxon test in order to obtain the same false alarm rate as DOMINO (Fig. 6).

On the second part of the simulation we implemented a misbehaving node shifting between one observation period acting normal and one observation pe-

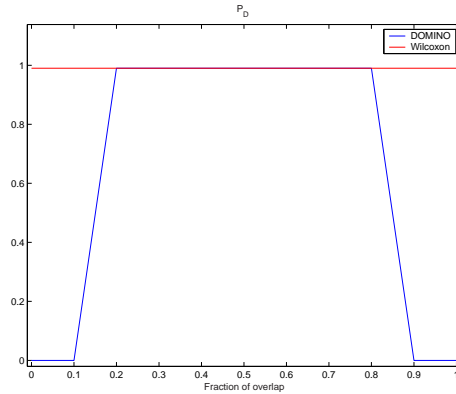


Figure 7: Probability of Detection versus the percentage of timing difference between the misbehaving node timing and the detection strategy

riod transmitting with zero backoffs. In Fig. 7 we plot the detection probability for the two tests versus the overlap between the timing of the monitoring node and the misbehaving node. For example a zero timing difference means the misbehaving node and the monitoring node start at the same time each period. In this case the selfish node would obtain the whole channel throughput during the zero back-off period while avoiding detection by DOMINO. However when the misbehaving node misses by twenty percent the timing interval, DOMINO is able to detect it, because the cheat count will increase in both periods, reaching the value K . A timing difference of one hundred percent means that the misbehaving node will again start his misbehavior at the beginning of a monitoring period for the monitoring node.

Although this scenario shows how standard nonparametric statistics are more resilient to intelligent strategies, in practice we believe that the fact that the adversary does not know the timing periods of DOMINO will be enough to deter him from misbehaving. On the other hand, another important question is the number of false alarms generated. One of the major problems with intrusion detection systems is the large number of false alarms, and with our selected parameters, six percent of false alarms might be too large a number to use in practice.

In the following section we take the first steps towards analyzing the false alarm rate produced by detection systems such as DOMINO.

6 Detection algorithm analysis

For each node's identity, the observation of the detection algorithm is a sequence of the node's backoffs X_1, X_2, \dots, X_n .

For detection, DOMINO has two parameters: $K \in \mathbb{N}$ and $\gamma \in (0, 1)$. These parameters can be tuned for the desired level of system performance, i.e. the trade-off between the probability of detection and the probability of false alarms. The detection algorithm is as follows:

cheat_count = 0

```

if           $\frac{1}{n} \sum_{i=1}^n X_i \leq \gamma B$ 
    then      cheat_count = cheat_count + 1
    if        cheat_count > K
        then    node misbehaving
    elseif   cheat_count > 0
        then    cheat_count = cheat_count - 1

```

In order to provide an analytical model for the performance of the algorithm, we model the detection mechanism by two parameters

1. $p := \Pr \left[\frac{1}{n} \sum_{i=1}^n X_i \leq \gamma B \right]$
2. A Markov chain with transition probabilities p and $1 - p$, where the absorbing state represents the case when the algorithm declares a misbehaving node. Note that we assume n is fixed, so p does not depend on how many backoffs we observed. This makes the detection mechanism to false alarms as we explain later in our analysis. A Markov chain with $K = 2$ is shown in Figure 8.

Assumptions:

We first consider the normal operation, where the monitored node is honest. With an honest node we assume each X_i is uniformly distributed in the set $\{0, 1, \dots, CW - 1\}$, therefore $\mathbf{E}[X_i] = \frac{CW-1}{2}$ and $\mathbf{Var}(X_i) = \frac{CW^2-1}{12}$. Furthermore let $B = \mathbf{E}[X_i] = \frac{CW-1}{2}$.

Note that assuming CW is the minimum contention window size, our analysis will give us a lower bound on the probability of false alarms.

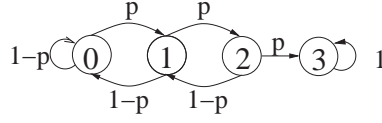


Figure 8: Markov Chain with four states, for $K=2$

6.1 Calculation of p

From the definition of p :

$$\begin{aligned}
 p &= \Pr \left[\sum_{i=1}^n X_i \leq n\gamma B \right] \\
 &= \sum_{k=0}^{\lfloor n\gamma B \rfloor} \Pr \left[\sum_{i=1}^n X_i = k \right] \\
 &= \sum_{k=0}^{\lfloor n\gamma B \rfloor} \sum_{\{(x_1, \dots, x_n) : \sum_{i=1}^n x_i = k\}} \frac{1}{CW^n}
 \end{aligned}$$

where the last equality follows from the fact that the X_i 's are independent and identically distributed with probability mass function $p(x_i) = 1/CW^n$ for all $x_i \in \{0, 1, \dots, CW - 1\}$.

Finding the number of ways that n integers can sum up to k is $\binom{n+k-1}{k}$ and $\sum_{k=0}^L \binom{n+k-1}{k} = \binom{n+L}{L}$. However we have a constraint in our problem, and it is the fact that X_i can only take values up to $CW - 1$, which will be in general smaller than k , and thus we cannot apply the above combinatorial formula. Furthermore a direct computation of all the ways x_i bounded integers sum up to k is very expensive. As an example let $CW = 32 = 2^5$ and let $n = 10$. Therefore a direct summation for finding p would need at least 2^{50} iterations.

Fortunately an efficient alternative way exists for computing p . Let

$$Y := \sum_{i=1}^n X_i$$

Then we know that the moment generating function of Y can be computed as fol-

lows:

$$\begin{aligned}
M_Y(s) &= M_X(s)^n \\
&= \frac{1}{CW^n} \left(1 + e^s + \dots + e^{(CW-1)s}\right)^n \\
&= \frac{1}{CW^n} \sum_{\substack{k_1, k_2, \dots, k_{CW} \\ \{k_1 + \dots + k_{CW} = n\}}} \binom{n}{k_1; \dots; k_{CW}} 1^{k_1} \dots e^{s(CW-1)k_{CW}}
\end{aligned}$$

where $\binom{n}{k_1; k_2; \dots; k_{CW}}$ is the multinomial coefficient $\frac{n!}{k_1! k_2! \dots k_{CW}!}$.

By comparing terms with the transform of $M_Y(s)$ we observe that $\Pr[Y = k]$ is the coefficient of the term e^{ks} in Equation 1.

As n grows larger, there is an approximate way of computing p . This approximation comes from the fact that as n increases, by the Central Limit Theorem, Y converges to a Gaussian random variable. Thus

$$p = \Pr[Y \leq \lfloor n\gamma B \rfloor] \approx \Phi(z)$$

where

$$z = \frac{\lfloor n\gamma B \rfloor - n \frac{CW-1}{2}}{\sqrt{(CW-1)(CW+1)n/12}}$$

and $\Phi(z)$ is the error function:

$$\Phi(z) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^z e^{-x^2/2} dx$$

In Figure 9 we show the exact and approximate calculation of p as a function of n , for the parameters $\gamma = 0.9$ and $CW = 32$. This shows that the approximation is very accurate even for small values of n . This suggests that in order to get some uniform results, we should monitor the backoff process X_1, \dots, X_i until i is equal to some specified n (e.g. 20), instead of monitoring for a fixed time (e.g. 10 seconds). The reasoning is because if we monitor for a fixed time, then a node accessing the channel only once during the monitoring time ($n = 1$) will have much higher chances of raising a false alarm than a node that accesses the channel several times in that period (e.g. $n = 30$).

6.2 False Alarm Rate

In order to compute the false alarm rate we need to find the expected time to absorption to Markov chains similar to the one shown in Figure 8. Let μ_i be the expected number of transitions until absorption, given that the process starts at state

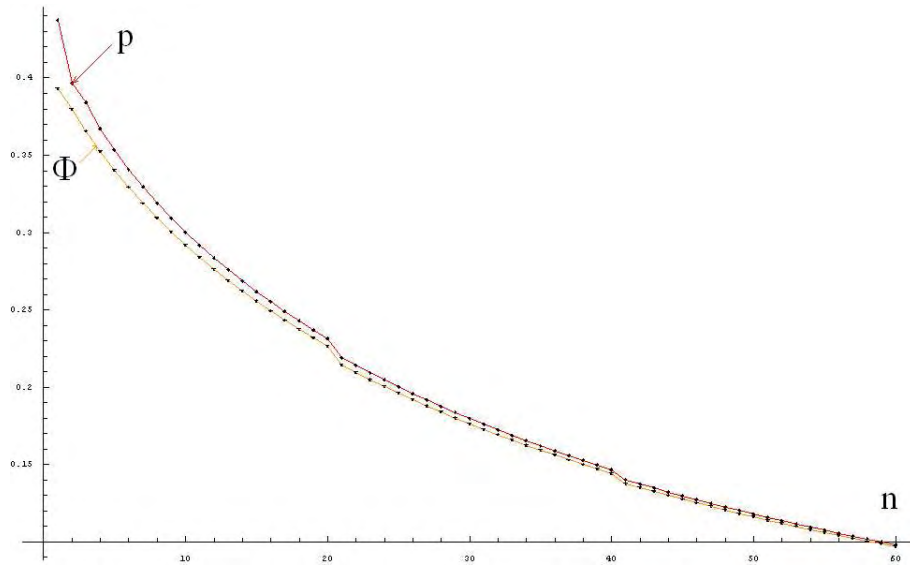


Figure 9: Exact (red) and approximate (orange) values for p as a function of n for $\gamma = 0.9$ and $CW = 32$.

i. Then, the expected times to absorption $\mu_0, \mu_1, \dots, \mu_{K+1}$ are the unique solution to the equations

$$\begin{aligned} \mu_{K+1} &= 0 \\ \mu_i &= 1 + \sum_{j=0}^{K+1} p_{ij} \mu_j \text{ for } i \in \{0, 1, \dots, K\} \end{aligned}$$

where p_{ij} is the transition probability from state i to state j . For any K , the equations can be represented in matrix form:

$$\begin{bmatrix} -p & p & 0 & 0 & \dots & 0 \\ 1-p & -1 & p & 0 & \dots & 0 \\ 0 & 1-p & -1 & p & 0 & 0 \\ 0 & 0 & 1-p & -1 & p & 0 \\ & & \vdots & & & \\ 0 & 0 & \dots & 0 & 1-p & -1 \end{bmatrix} \begin{bmatrix} \mu_0 \\ \mu_1 \\ \mu_2 \\ \vdots \\ \mu_K \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \\ -1 \\ \vdots \\ -1 \end{bmatrix}$$

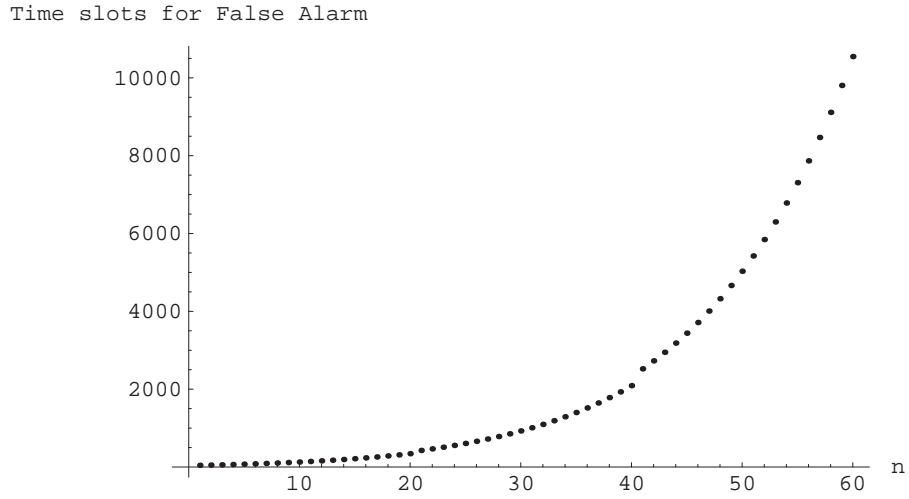


Figure 10: Expected time slots for a false alarm as a function of n for $\gamma = 0.9$ and $CW = 32$.

So for example for $K = 3$

$$\begin{bmatrix} -p & p & 0 & 0 \\ 1-p & -1 & p & 0 \\ 0 & 1-p & -1 & p \\ 0 & 0 & 1-p & -1 \end{bmatrix} \begin{bmatrix} \mu_0 \\ \mu_1 \\ \mu_2 \\ \mu_3 \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \\ -1 \\ -1 \end{bmatrix}$$

and the solution we are interested is

$$\mathbf{E}[\text{time to false alarm}] = \mu_0 = \frac{1-p+2p^2+2p^3}{p^4}$$

In Figure 10 we show the expected number of time slots (where each time slot is given by n observations) for raising a false alarm.

The specific n in our observation period depends on the network usage, traffic characteristics and number of users contending for the channel. In future work we will provide a detailed analysis of the expected number of false alarms given these network characteristics and implementation validation for the results.

7 Conclusions and Future Work

Misbehavior at the MAC layer by changing the backoff mechanism can lead to performance degradation and even denial of service attacks in ad hoc networks. In this paper we have presented ERA-802.11 in order to help in the detection of non-colluding selfish nodes. However, even when neighboring nodes know the backoff time agreed by a sender, the network topology, hidden nodes, the exponential backoff due to the capture effect, and network traffic characteristics can severely degrade the correct detection of a misbehaving node. We plan to investigate in future work how the detection accuracy of the monitoring nodes perform with respect to the number of hidden nodes in their neighborhood and to different types of network traffic.

For colluding nodes, the problem of detecting backoff manipulation at the MAC layer becomes very difficult. Besides the same detection difficulties that we have for a non-colluding node, we are now required to take several samples of the backoff time in order to come up with an accurate decision. In future work we will focus on a more rigorous treatment of the detection problem and show under the consideration of parameters such as network topology, mobility and traffic characteristics, the difficulty or feasibility of the problem.

Finally, we assumed in this paper that there was a reputation algorithm receiving our detection results. There is still the open question of how to react when we detect a misbehaving node. How bad is the performance degradation for the rest of the network? What is the best punishment strategy? It is our view that the reputation mechanism should have a layered security mechanism in order to provide an educated decision on how to react to MAC layer misbehavior.

Acknowledgements

We would like to thank Prof. Jonathan Katz for helpful discussions on Coin Flipping over 802.11.

This material is based upon work supported by the U.S. Army Research Office under Award No. DAAD19-01-1-0494 to the University of Maryland College Park.

References

- [1] E. Altman, R. E. Azouzi, and T. Jimenes, "Slotted aloha as a stochastic game with partial information," in *Proceedings of WiOpt*, 2002.

- [2] J. Bellardo and S. Savage, "802.11 denial-of-service attacks: Real vulnerabilities and practical solutions," in *Proceedings of the USENIX Security Symposium*, Washington D.C., August 2003.
- [3] M. Blum, "Coin flipping by telephone: a protocol for solving impossible problems," in *Proceedings of the 24th IEEE Spring Computer Conference, COMPCON*, 1982, pp. 133–137.
- [4] S. Buchegger and J. Y. Le Boudec, "Nodes bearing grudges: Towards routing security, fairness, and robustness in mobile ad hoc networks," in *Proceedings of Tenth Euromicro PDP (Parallel, Distributed and Network-based Processing)*, Gran Canaria, January 2002, pp. 403 – 410.
- [5] S. Buchegger and J.-Y. Le Boudec, "Performance analysis of the confidant protocol," in *Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing*, 2002, pp. 226–236.
- [6] M. Cagalj, S. Ganeriwal, I. Aad, and J.-P. Hubaux, "On cheating in csma/ca ad hoc networks," EPFL, Tech. Rep., February 2004.
- [7] A. Cardenas, S. Radosavac, and J. Baras, "Detection and Prevention of MAC layer misbehavior in ad hoc networks," in *Proceedings of the 2nd ACM workshop on security of ad hoc and sensor networks*, 2004.
- [8] J. R. Douceur, "The Sybil attack," in *Proceedings of the 1st International Peer To Peer Systems Workshop (IPTPS 2002)*, March 2002.
- [9] V. Gupta, S. Krishnamurthy, and M. Faloutsos, "Denial of service attacks at the mac layer in wireless ad hoc networks," in *Proc IEEE MILCOM*, October 7-10, 2002.
- [10] J. Hájek, Z. Šidák, and P. Sen, *Theory of rank tests*. Academic Press, New York, 1999.
- [11] P. Kyasanur and N. Vaidya, "Detection and handling of mac layer misbehavior in wireless networks," in *Proceedings of the International Conference on Dependable Systems and Networks*, June 2003.
- [12] A. B. MacKenzie and S. B. Wicker, "Stability of multipacket slotted aloha with selfish users and perfect information," in *Proceedings of the IEEE INFOCOM*, 2003.
- [13] M. Raya, J.-P. Hubaux, and I. Aad, "Domino: A system to detect greedy behavior in ieee 802.11 hotspots," in *Proceedings of the Second International*

Conference on Mobile Systems, Applications and Services (MobiSys2004),
Boston, Massachussets, June 2004.