Detection and Prevention of MAC Layer Misbehavior for Ad Hoc Networks

Alvaro A. Cardenas, 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

Abstract It is known that 802.11 is not resilient to selfish behavior and denial of service (DoS) attacks targeting the MAC layer. In this study, we assess the performance of the CSMA/CA scheme and investigate its efficiency with regard to security and information assurance in mobile ad hoc wireless networks. We investigate several variants of selfish node behaviors that abuse the random choice of Contention Window in the 802.11 DCF MAC protocol. We show that selfish behavior in the MAC layer can have devastating side effects on the performance of wireless networks, similar to the effect of DoS attacks. Our main contributions are the prevention and detection of backoff manipulation in an ad hoc network. First, we propose an algorithm to prevent backoff manipulation in a communication link between a sender and a receiver when at least one of the nodes behaves honestly. Secondly we introduce algorithms for detection of backoff manipulation by a pair of colluding nodes.

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 will obey the specification. However when this protocols are implemented in an untrusted environment, nodes can misbehave to obtain a given goal. A selfish user for example can change the congestion avoidance parameters of TCP, mainly increase the slope of the congestion window linear growth and decrease the congestion threshold multiplicative reduction, in order to obtain unfair advantage over the rest of the nodes in the network [1]. A selfish user can also disobey the rules to access the wireless channel in order to obtain a higher throughput than the other nodes. In limited power devices, certain nodes might refuse to forward packets in behalf of other sources. Misbehaving in the network protocols will degrade the performance of the network as experienced by the honest participants. To fully address this problem a layered security mechanism should be deployed in order to penalize or force to cooperate misbehaving nodes that degrade the performance seen by honest participants. In this paper our goal is to protect the IEEE 802.11 MAC layer from unfairness and collisions of packets caused by selfish users in ad hoc networks.

The MAC layer in a communications 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. 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 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 improve performance of its products.

The IEEE 802.11 Medium Access Control (MAC) protocol uses a distributed contention resolution mechanism for sharing the wireless channel and its design tries to ensure a relatively fair access to the medium for all participants of the protocol. The MAC layer has mechanisms to protect itself from congestions, but these mechanisms can be abused by attackers and used to disrupt communication in the MAC layer. In 802.11 the nodes follow binary exponential backoff scheme that favors the last winner amongst the competing nodes. Even when all contending nodes are well behaved this mechanism can lead to the capture effect where nodes that are heavily loaded tend to capture the channel by continuously transmitting data which makes lightly loaded neighbors to back off continuously. Very similar effects are obtained when one of the contending nodes is selfish.

A selfish user keeps the channel busy in order to maximize its own throughput. As a side effect of this behavior, regular nodes cannot use the channel for transmissions, which leads to a denial of service (DoS) attack [15]. Due to the randomness introduced in the choice of the backoff, it is difficult to detect when a selfish user has chosen small values of the backoff by chance or not. Selecting small backoff values gives him an advantage over the other contending nodes that uniformly choose the backoff time.

A selfish node can cause congestion in the network as a side effect of maximizing its throughput. It can generate an excessive amount of traffic [6, 15] or specific traffic patterns that prevent certain nodes from accessing the medium and, therefore, maximizing its throughput.

In this work we investigate selfish behavior achieved by manipulating the backoff mechanism of IEEE 802.11 MAC protocol. However, a selfish user can implement a whole range of strategies to maximize its access to the medium. The most likely strategy that an intelligent selfish user will employ is to use different schemes of manipulating the rules of the MAC layer. 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) etc. A successful detection scheme should take into account all possible cheating schemes in the MAC layer and detect both users that employ only one scheme and users that employ a combination of several schemes (i.e. first choosing small backoff values, then assigning large NAV values to its neighbors etc.)

In Section 4 we attempt to solve the problem of misbehavior by preventing cheating in the backoff stage of IEEE 802.11 for non-colluding nodes. In Section 5 we solve the problem of detecting misbehavior of colluding nodes. Both of the protocols can be extended for any probabilistic distributed protocol to access the channel.

2 Related Work

Selfish misbehavior at the MAC layer has been addressed mostly from a game theoretic perspective. The goal in a game theoretic setting is to design distributed protocols that guarantee the existence, uniqueness and convergence to a Nash equilibrium. 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, a cost for every node for each time it access the channel has to be included. For example in [11, 8], the selfish users in Aloha attempt to maximize their throughput minus a cost for accessing the channel (e.g. energy consumption). Another game theoretic scheme for CSMA/CA schemes is presented in [12]. It shows how a Nash equilibrium is achieved against 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 output 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 to avoid the hidden terminal problem, so this scheme is mostly intended for wireless LANs. Game theoretic protocols assume all nodes are selfish (the worst case scenario) and therefore the throughput achieved by honest nodes in these protocols is substantially less than in protocols where the honest majority cooperates.

Detection of misbehavior at the MAC layer has received little attention in the literature. 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, based on comparing average values of the backoff to given thresholds is suboptimal for every strategy of the greedy user. In this paper we propose a new detection schemes for the backoff manipulation that we believe will improve the performance of systems such as DOMINO.

Kyasanur and Vaidya [10] propose a modification to 802.11 for facilitating the detection of misbehaving nodes. 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 three parts:

- Detection: The sender deviates from the protocol if the observed number of idle slot B_{act} is smaller than a specified fraction α of the assigned backoff B_{exp}. To alleviate for false positives in the presence of a hidden terminal, the sender will count only as busy time slots those slots not reserved by other nodes with a RTS/CTS.
- Penalty: Assign a penalty with a measure of deviation $D = \max(\alpha B_{exp} - B_{act}, 0)$.
- Diagnosis Scheme: 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 this paper we extend the idea of [10] by presenting an algorithm that ensures a honest backoff selection among the sender and a receiver as long as one of the participants is honest.

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 [7].

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).

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

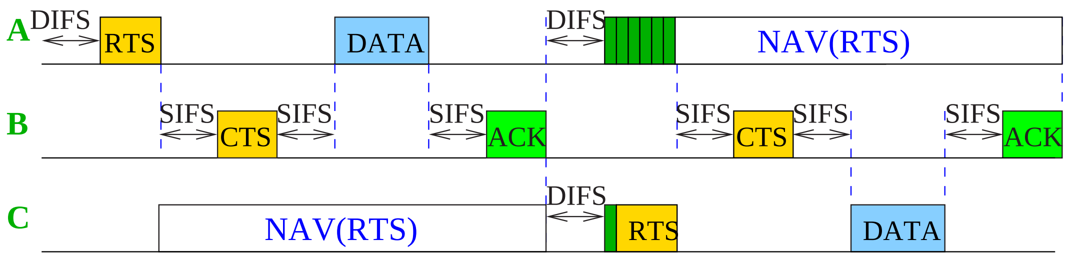

Figure 1: Nodes A and C contend for accessing node B. The first time A reserves the channel, and in the second time C accesses the channel.

chosen uniformly from the set $\{0, 1, \ldots, CW-1\}$ (CW is the contention window size), and transmits after waiting for b idle slots. Nodes exchange RTS and 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, which includes 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 transmission is successful the host resets its CW to a minimum value CW_{min}.

The following diagram depicts the handshake mechanism used in 802.11 DCF

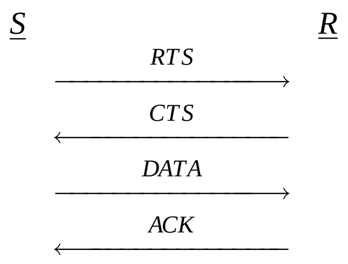

Figure 1 shows an example of contending nodes using the protocol.

The timers are selected based on which physical layer 802.11 is using. For example table 1 shows the parameters used when the physical layer is using direct sequence spread spectrum (DSSS) (timeouts are not defined in the IEEE 802.11 standard, but are usually set to the values given in the table).

The exchange of RTS/CTS control frames is an optional feature of 802.11. The primary reason for enabling this functionality is if you expect frequent collisions or retransmissions due to hidden nodes. If RTS/CTS is not enabled then for any data packet transmitted only two frames are needed: $DATA/ACK$. Since hidden nodes are common in ad hoc networks, we assume the exchange of RTS/CTS is active.

Cheating in the 802.11 protocol can be done in many places, for example by setting large

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

Table 1: Parameters for DSSS

NAV values for other contending nodes, by deauthenticating neighboring nodes, or by violating the backoff mechanism. However most of the cheating is fairly easy to detect or defend except for backoff manipulation [13, 2]. In this paper we propose in section 4 a technique for ensuring honest backoffs when there is at most one cheating node and in section 5 we consider detection tests when there are colluding nodes in the network.

4 ERA: Ensuring Randomness for 802.11

One of the main assumptions used in the scheme proposed in [10] is that the receiver has 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. For example, the receiver can be expecting data from a particular sender and thus assign small backoff values to it, degrading the performance of the other neighboring senders. The receiver can also assign large backoff values to some neighbors and deny them access to the network. Furthermore, if there are two pairs of receivers and senders in the same range, a malicious receiver can overhear the backoff value assigned by the other receiver to the first sender, and then unilaterally select a random backoff for the second sender such that it will cause a collision between the two senders. This scenario can be seen in figure 2.

In this section we propose an extension to the 802.11 CSMA/CA protocol that will ensure a uniformly distributed random backoff, in the case that at least one of the parties is honest. The basic idea follows the protocol for flipping coins over the telephone by Blum [3].

The protocol can be embedded in 802.11 to be 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:

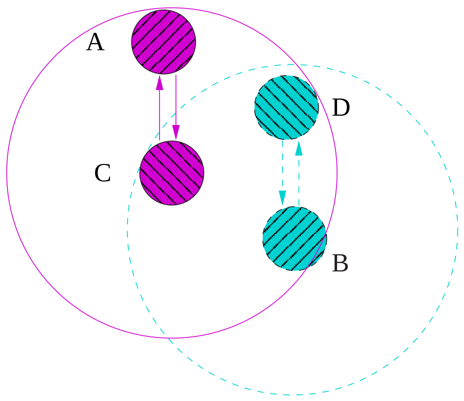

Figure 2: Node C transmits to A and node B wants to transmit to D. After hearing the backoff assigned by A to C, node D will assign a backoff to node B such that it will collide with C.

$$
\begin{array}{lcl}
\underline{S} & & \underline{R} \\
n \leftarrow \{0,1\}^{k_{nonce}} & \xrightarrow{RTS||n} & r \leftarrow \{0,1,\ldots,CW_{min}-1\} \\
 & \xleftarrow{CTS||\sigma} & \sigma = \mathsf{Commit}(\mathsf{r}||\mathsf{n}) \\
r' \leftarrow \{0,1,\ldots,CW_{min}-1\} & \xrightarrow{DATA||r'} & \sigma' = \mathsf{Open}(\sigma) \\
 & \xleftarrow{ACK||\sigma'} & \\
\mathsf{Commit}(\mathsf{r}||\mathsf{n}) \overset{?}{=} \sigma & & \\
b_i = r_i \oplus r'_i \text{ for } i \in \{1,\ldots,m\} & & b_i = r_i \oplus r'_i \text{ for } i \in \{1,\ldots,m\}
\end{array}
$$

We now explain the protocol step by step.

1. In the first step the sender S selects a nonce: a number n selected uniformly randomly from the set $\{0,1,\ldots,2^{k_{nonce}}\}$, denoted as $n \leftarrow \{0,1\}^{k_{nonce}}$. k_{nonce} is a security parameter selected so that it is difficult for R to guess the nonce, for example $k_{nonce} = 64$. This step is optional and is done in order to prevent an offline attack to the commitment scheme.

2. In the second step the receiver R selects a random backoff r from the set $\{0,1,\ldots,CW_{min}-1\}$ and commits to it. In binary notation r is a random bit string of length m ($r = r_1r_2\cdots r_m$), where $m = \log_2 CW_{min}$ (note that the minimum contention window size CW_{min} is always a power of two). The commitment scheme Commit is such that the following two properties are satisfied: (in a reasonable amount of time- time outs in 802.11 for reserving the channel are between $300\mu s$ to $350\mu s$):

   Binding: After sending $\mathsf{Commit}(\mathsf{r}||\mathsf{n})$, 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 $\mathsf{Commit}(\mathsf{r}||\mathsf{n})$, 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 $r' = r'_1 r'_2 \cdots r'_m$ from $\{0, 1, \ldots, CW_{min} - 1\}$.

4. Finally R opens its commitment to S. Opening a commitment is an operation that reveals to S the committed value r, plus some information so that the other party can confirm that the value revealed is indeed the value that was committed. If the value opened by the R is correct, then the sender and the receiver, compute the backoff $b = b_1 b_2 \cdots b_m$ as the xor of the bits: $b_i = r_i \oplus r'_i$. If it is not, then the sender can report a misbehavior to the reputation management system we assumed.

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 see that the following commitment scheme satisfies the binding (by assuming H to be collision resistant) and hiding properties (by assuming H is a random oracle) required:

$$\underline{\mathsf{Commit}(\mathsf{r}||\mathsf{n})}$$

$$i \leftarrow \{0,1\}^k$$
$$\mathsf{Output} = \mathsf{H}(\mathsf{i}||\mathsf{r}||\mathsf{n})$$

$$\underline{\mathsf{Open}(\mathsf{H}(\mathsf{i}||\mathsf{r}||\mathsf{n}))}$$

$$\mathsf{Output} = (\mathsf{i}, \mathsf{r})$$

where k is a security parameter (e.g. $k = 64$, 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 r as well as i, so that S can check if the commitment was valid.

For a practical case, consider when the physical layer of 802.11 is set to direct sequence spread spectrum (DSSS). 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 would incur with a reasonable extra $k + m = 69$ bits. If we use SHA-1 to implement the hash function of the commitment then we obtain a message digest of 160 bits. The *RTS* frame would be doubled in size if we use the full message digest, however 320 bits would still be insignificant to a normal *DATA* frame which can be up to 2312 bytes, therefore the use reservation by exchanging RTS/CTS would still be more effective than the basic access mode of 802.11 DCF (no RTS/CTS exchange). If doubling the size of a *RTS* 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. If $k_{nonce} = 0$, an attacker could attempt to find collisions offline and then open a false commitment. There is a tradeoff then between the length of the message digest used for the commitment and the security parameter k_{nonce}. The longer the message digest, the less likely there is an offline attack.

If we remove the nonce from the protocol we can make the sender use more transmission power than the receiver with the following protocol:

$$\begin{array}{lcl}
\underline{S} & & \underline{R} \\
r \leftarrow \{0,1,\ldots,CW_{min}-1\} & & \\
\sigma = \mathsf{Commit(r)} & \xrightarrow{\qquad RTS||\sigma \qquad} & \\
 & \xleftarrow{\qquad CTS||r' \qquad} & r' \leftarrow \{0,1,\ldots,CW_{min}-1\} \\
 & \xrightarrow{\quad DATA||\mathsf{Open}(\sigma) \quad} & \mathsf{Commit(r)} \overset{?}{=} \sigma \\
b_i = r_i \oplus r'_i & \xleftarrow{\qquad ACK \qquad} & b_i = r_i \oplus r'_i
\end{array}$$

in this case, the receiver will only be required to append an extra five bits to the CTS frame. A data frame would be appended b and r which sum up to only 69 bits, which is still very small compared to the payload. So our only major modification to 802.11 in terms of communication bandwith used is the payload of the message digest σ.

To finalize this section we just point out that once a sender and a receiver have agreed on a given random backoff the receiver can monitor the behavior of the sender and report a misbehaving node to the reputation management system. To detect if the sender deviated from the agreed backoff, we can use the detection algorithm given in [10], in particular we note that if the sender sensed the channel to be idle it would count down its backoff timer, however the receiver can sense the channel to be busy (channel reserved by a hidden node to the sender) and under normal operation it would not count down its timer. In order to avoid false positives created by the hidden terminal unit, the receiver will always decrement the counter that monitors the sender. It is also mentioned in [10] 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 $\{0,1,\ldots,(CW_{min}+1)2^{i-1}-1\}$ (where i is the number of retransmissions and it keeps increasing until the size of the contention window reaches CW_{max}).

5 Detection System

The previous algorithm is not resistant to colluding nodes selecting small backoff values with the intention of maximizing their throughput, at the expense of throughput degradation for neighboring nodes. Similarly, colluding nodes can agree on sharing large backoff times, causing large delays in their transmissions so that they are rejected from inclusion in routing layer paths. In the rest of this section we will focus only on detecting colluding nodes with the intention of selecting small backoff values. Detecting large backoff times can be done in the similarly.

When the sender and the receiver collude by selecting their random numbers for the contention window a priori, they can deny access to the network. For example, consider figure 3 and assume C

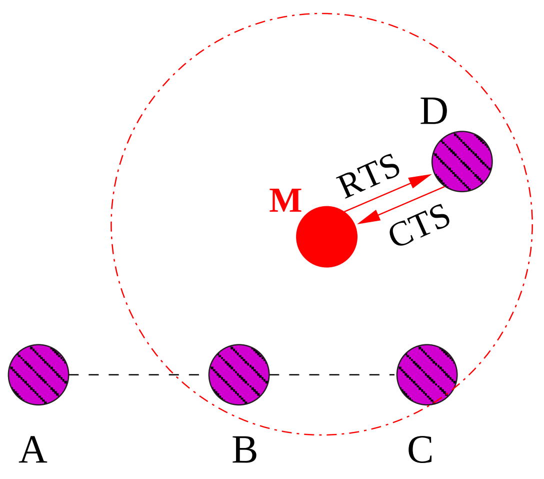

Figure 3: Nodes M and D collude to and interfere in the communication path of nodes B and C

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 exactly the backoff time 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 a priori their numbers. For example they can collude to present to node C a valid message agreement on the backoff zero by selecting the following values:

$\underline{M}$		$\underline{D}$
$n \leftarrow \{0,1\}^{k_{nonce}}$	$\xrightarrow{RTS\|\|n}$	
	$\xleftarrow{CTS\|\|\sigma}$	$\sigma = \mathsf{Commit}(00000\|\|n)$
	$\xrightarrow{DATA\|\|00000}$	$\sigma' = \mathsf{Open}(\sigma)$
	$\xleftarrow{ACK\|\|\sigma'}$	
$b = 00000$		$b = 00000$

At the end, the sender will not wait for any time slot in the backoff period and will send immediately its last $DIFS$ ends. In Figure 4 we show how the sequence of small backoffs $0, 1, 2, \ldots$ from node M will cause the timer for the CTS frame of node A to time out. Node A will therefore backoff exponentially repeatedly, making it less likely to access the network. We implemented this setting in the network simulator Opnet, and allowed the colluding nodes to transmit with the smallest backoff in a period of time. In figure 5 it is shown how the colluding nodes denied access to the network to node A during that period.

Having motivated the need to detect colluding MAC layer misbehavior in ad hoc networks, in

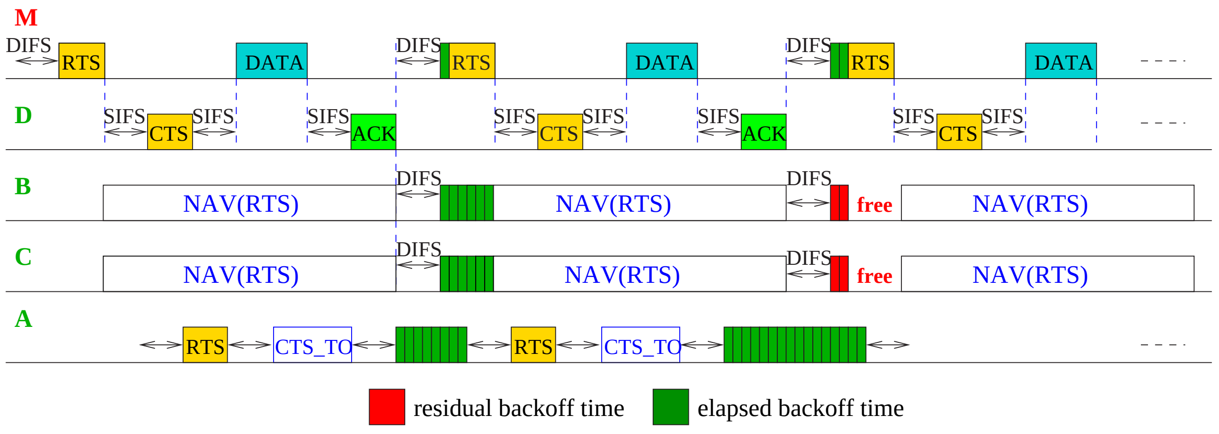

Figure 4: Nodes M and D collude and select a very small backoff, thereby denying access to node A by causing CTS timeouts.

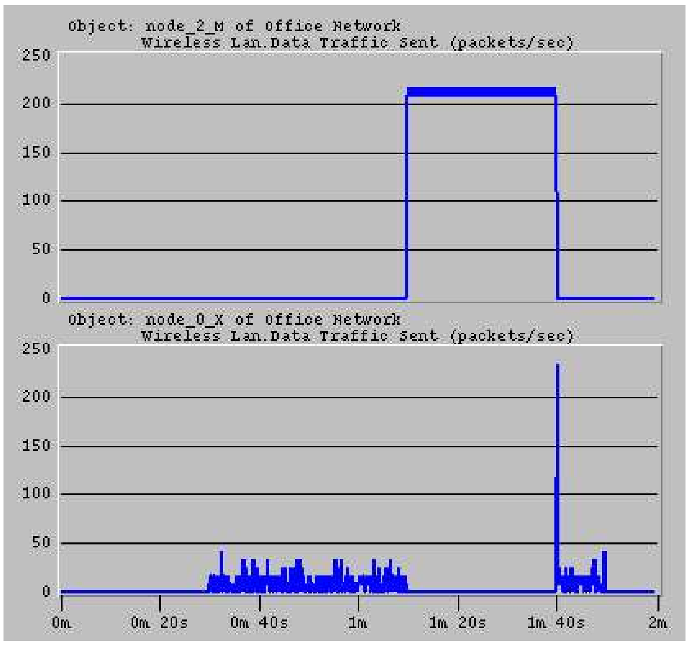

Figure 5: Simulation of traffic sent by node M (top figure) versus traffic sent by node B (top figure). When D and M collude A is denied access to the network.

the rest of this section we focus on designing algorithms such that neighboring nodes can detect when other nodes do not follow a uniformly random backoff time. Although our emphasis is on ad hoc networks, the same algorithms can be applied to other settings, for example to monitor greedy behavior at access points in WLANs using the original 802.11 protocol (802.11 without the modifications introduced in the previous section).

5.1 Test for backoff manipulation

A misbehaving host that knows there are neighboring nodes monitoring its backoff period will try to avoid detection while maintaining a high throughput. It is difficult however to come up with a strategy that the misbehaving nodes will use to access the channel more frequently. If we assume for example that the misbehaving nodes will select their backoff uniformly random from the set $\{0,1,\ldots,CW_{min}/4\}$ and monitor the access times for no occurrences above $CW_{min}/4$, then an adaptive misbehaving node will override our detection mechanism by selecting a value above $CW_{min}/4$ before our test makes a decision. Selecting the time at which to wait for a backoff larger than $CW_{min}/4$ assumes the colluding nodes know perfectly well our assumptions about the attack and the parameters used for its detection. Giving the misbehaving nodes the capability to know our assumptions on the attack allow us to consider detection schemes which do not assume that misbehaving nodes will be systematically fooled. In general, we assume that the colluding nodes are intelligent (they know everything each monitoring node knows about the detection scheme), and that they can make inferences about the situation in the same way as the monitoring nodes can.

In general, the more inaccurate assumptions we make about the attack, the less effective our detection algorithm is. In particular we do not know a priori the probability distribution that the colluding nodes will use in order to access the channel.

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 penalize also 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}.

Before presenting the statistical tests, let us introduce some notation. 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 $X_1,\cdots,X_n$. We assume we make a decision after n observations.

5.1.1 Tests for change in the mean

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) and then, 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 K will only flag a false alarm after the statistic X_{ac} has exceeded γB_{min} K times. They tested the detection scheme against a misbehaving node whose strategy was to select backoff values uniformly random from the set $\{0,1,\ldots,\lfloor CW_{min} \times \delta \rfloor\}$, where δ $(0 \leq \delta \leq 1)$ represents the amount of misbehavior, with $\delta = 0$ meaning that the station transmits without backoff and with $\delta = 1$ 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.

This algorithm in DOMINO however is not optimal for more intelligent strategies of the misbehaving nodes. A misbehaving node in 802.11 (or a pair of colluding nodes in ERA-802.11) will avoid detection and still achieve access to the channel more than half of the contending trials by selecting the following backoff scheme:

For the first $(K-1)n$ backoffs, select a backoff of zero (this exploits the fact that an alarm is never raised until the mean statistic has exceeded γB_{min} K times.) Then alternate between a backoff of zero and the backoff $2\gamma B_{min}$.

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 no matter how many other contending neighbors there are. Since in an ad hoc network there might be several nodes monitoring the misbehaving backoff strategy, each with different values of K (different times of arrivals to the neighborhood), a safe strategy is just to alternate between 0 and $2\gamma B_{min}$, a strategy which still gives access to the channel to the misbehaving nodes more than half of the times.

This same fate is shared with several nonparametric tests measuring how "shifted" in the X axis is the alternative distribution from the null distribution (which is assumed to be symmetric at zero). If we define the independent and identically distributed (i.i.d.) random process: $Y_1,\ldots,Y_n$ by: $Y_i = B_{min} - X_i$ we can use a Sign Test or a Wilcoxon Test [9].

5.1.2 Entropy Estimation

Clearly if we could test how random the backoff mechanism of the misbehaving node is, we would make the evasion of detection more difficult. However applying empirical statistical tests to random number generators is a highly heuristic affair. Furthermore an adversary can still cheat several of the tests, such as cumulative sums, frequency counts and tests based on random walks, while still accessing the channel much more often than honest participants. We therefore turn our attention to the entropy of the backoff process, as one of the more used measures of randomness. Note however that we do not need to measure the entropy of the backoff values observed versus the entropy of a uniform random variable with range $\{0,1,\ldots,CW_{min}-1\}$. Doing this would cause much more false alarms due to the large parameter space for estimation (estimating more parameters means our observations before making a decision would increase dramatically). Therefore we partition the backoff range into M bins, where $2 < M < CW_{min}$ so that the estimation of the probabilities in each bin is more efficient for small sample sizes, while limiting the attacker strategies. As a statistical test, we measure the empirical entropy of the M bins (which ideally should be close to $-\log_2 \frac{1}{M}$) with a threshold given by a fixed false alarm rate.

In order to evaluated the performance of the entropy statistic, we compare it to the detection accuracy of DOMINO, and a Wilcoxon nonparametric rank test. The strategy of the selfish user is to fool the detection scheme by alternating backoff values between 0 and α, for $\alpha = 0,\ldots,CW_{min}-1$. α therefore is a measure of evasion by the misbehaving node where $\alpha = 0$ means the node transmits without any backoff, and $\alpha = CW_{min}-1$ means the node alternates between transmitting without any backoff and backing off for CW_{min}. The parameters of the simulation were 10^4 samples, $n = 20$, $CW_{min} = 32$ and the number of bins to evaluate the entropy was $M = 8$. Figure 5 shows how the

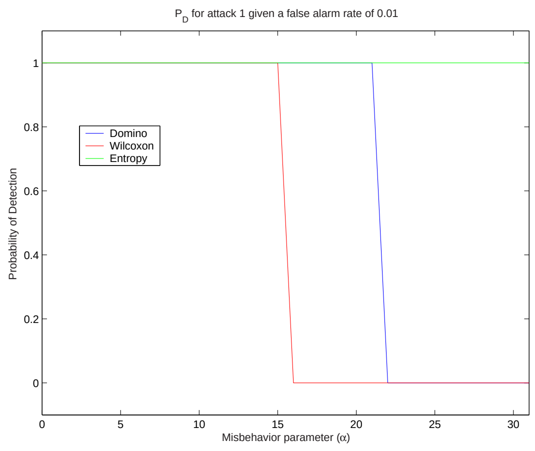

Figure 6: Entropy Test Limits the Strategy Space of the Misbehaving Nodes

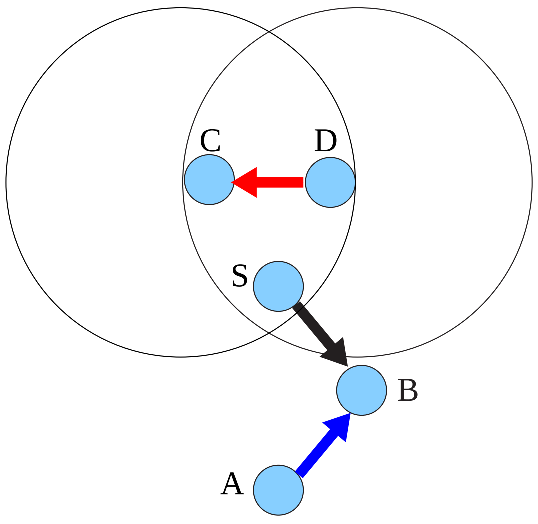

Figure 7: Due to asymmetrical network topology, node B cannot guarantee fair access to it among nodes A and S

entropy test can easily detect the selfishness while the Wilcoxon test misses the strategy when $\alpha > B_{min}/2$ and DOMINO when $\alpha > \gamma B_{min}$. This result should not be a surprise, as the entropy of the selfish behavior in this case is just 2. Note that in the case of 8 bins, the optimal strategy for the misbehaving node is to select the smallest backoff within each interval: 0,3,7,11,15,19,23,27. Furthermore if the selfish node knows the false alarm rate, then it can do better by selecting just the first values: 0,3,7,11,15,19 (when we are only tolerating up to an entropy of $\log_2 6$).

5.2 Test for Unfair Use

If we want to be conservative in the detection of misbehavior, after we obtain an alarm from the backoff manipulation algorithm, we can correlate the identity of the accused node, with the actual throughput difference among contending nodes. However, the actual throughput depend on many variables that are beyond the reach of the monitoring node. It depends on the traffic pattern of the network (other nodes might not have anything to send), and the network topology: figure 7.

Since 802.11's MAC layer has been shown to be unfair at short and long term use, for arbitrary topologies in ad hoc networks [14], we do not attempt to test for fairness but rather we test for the amount of unfairness that we will be able to accept. This test is just to ensure that if the backoff manipulation test detects a misbehaving node, it is indeed making an unfair use of the network by diminishing the throughput experienced by neighboring nodes. In this setting we assume the monitoring nodes are contending to access the same receiver.\

In this model we assume the selfish user will only permit access to the medium to its N neighbors with only ε probability. Alternatively it can be seen as the amount of unfairness we allow.

On the other hand, a fair protocol would allow access to the medium proportional to the other nodes (when the nodes are in saturation condition). Therefore if we let L_R be the number of times we observe any node other than node A accessing the medium and L_A the number of times A accesses the medium, a simple likelihood ratio test to inform if A is being overly unfair or closer to a complete fair protocol is the following:

$$H_1 : \frac{(1-\varepsilon)^{L_A}\varepsilon^{L_R}}{(\frac{1}{N})^{L_A}(\frac{N-1}{N})^{L_R}} > 1$$

We stress out again that this test should not be used alone given the characteristic unfairness of 802.11, but rather as a way to complement backoff manipulation detection algorithms.

6 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 misbehaving node, the network topology and hidden nodes can severely degrade the correct detection of the node. We plan to investigate in the future how an implementation of ERA-802.11 performs with respect to the number of hidden nodes in the neighborhood. We also plan to investigate if the small overhead included in the reservation packets influences the throughput.

When we have colluding nodes the problem of detecting backoff manipulation at the MAC layer becomes very difficult. Not only are we required to take several samples in order to come up with an accurate decision, but the hidden node problem, the exponential backoff due to the capture effect, and the different topologies influence our 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, 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] A. Akella, S. Seshan, R. Karp, S. Shenker, and C. Papadimitriou, “Selfish behavior and stability of the internet: a game-theoretic analysis of tcp,” in *Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communications*, 2002, pp. 117–130.

[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] A. M. C. Barrett, M. Drozda and M. V. Marathe, “Analyzing interaction between network protocols, topology and traffic in wireless radio networks,” in *Proc. IEEE Wireless Communications and Networking Conference (WCNC’03)*, 2003.

[7] J. R. Douceur, “The sybil attack,” in *Proceedings of the 1st International Peer To Peer Systems Workshop (IPTPS 2002)*, March 2002.

[8] R. E. A. E. Altman and T. Jimenes, “Slotted aloha as a stochastic game with partial information,” in *Proceedings of WiOpt*, 2002.

[9] Z. S. Jaroslav Hajek and P. K. Sen, *Theory of Rank Tests*, second edition ed. Academic Press, 1999.

[10] 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.

[11] 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.

[12] I. A. Mario Cagalj, Saurabh Ganeriwal and J.-P. Hubaux, “On cheating in csma/ca ad hoc networks,” EPFL, Tech. Rep., February 2004.

[13] J.-P. H. Maxim Raya 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.

[14] X. G. Thyagarajan Nandagopal, Tae-Eun Kim and V. Bharghavan, “Achieving mac layer fairness in wireless packet networks,” in *Proceedings of the 6th annual international conference on Mobile computing and networking*, 2000, pp. 87–98.

[15] M. F. V. Gupta, S. Krishnamurthy, “Denial of service attacks at the mac layer in wireless ad hoc networks,” in *Proc IEEE MILCOM*, October 7-10, 2002.