# DOMINO: A System to Detect Greedy Behavior in IEEE 802.11 Hotspots *

Maxim Raya, Jean-Pierre Hubaux, Imad Aad
Laboratory for computer Communications and Applications(LCA)
School of Computer and Communication Sciences
EPFL, Switzerland
{maxim.raya, jean-pierre.hubaux, imad.aad}@epfl.ch

## ABSTRACT

The proliferation of hotspots based on IEEE 802.11 wireless LANs brings the promise of seamless Internet access from a large number of public locations. However, as the number of users soars, so does the risk of possible misbehavior; to protect themselves, wireless ISPs already make use of a number of security mechanisms, and require mobile stations to authenticate themselves at the Access Points (APs). However, IEEE 802.11 works properly only if the stations also respect the MAC protocol. We show in this paper that a greedy user can substantially increase his share of bandwidth, at the expense of the other users, by slightly modifying the driver of his network adapter. We explain how easily this can be performed, in particular with the new generation of adapters. We then present DOMINO (System for Detection Of greedy behavior in the MAC layer of IEEE 802.11 public NetwOrks), a piece of software to be installed in the Access Point. DOMINO can detect and identify greedy stations, without requiring any modification of the standard protocol at the AP and without revealing its own presence. We illustrate these concepts by simulation results and by the description of our prototype.

## Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General—Security and protection; C.2.3 [**Computer-Communication Networks**]: Network Operations—Network monitoring, Public networks; C.2.5 [**Computer-Communication Networks**]: Local and Wide-Area Networks.

*This work is partially funded by the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center supported by the Swiss National Science Foundation under grant number 5005-67322 (http://www.terminodes.org).


*MobiSYS'04,* June 6–9, 2004, Boston, Massachusetts, USA.


## General Terms

Design, Experimentation, Performance

## Keywords

MAC, IEEE 802.11, misbehavior, wireless LAN, hotspot, WISP

## 1. INTRODUCTION

The last few years were marked by the widespread deployment of IEEE 802.11 [3] hotspots that provide public wireless access to the Internet. This trend will continue in the near future, according to the predictions of the research firm Allied Business Intelligence (ABI) [23] that estimate that the revenue from hotspots will increase by up to 121% in the next five years and the number of hotspots will jump from 28,000 today to 160,000 by 2007. The commercial operation of these networks has already revealed a set of problems, such as security and billing, which are typically less important or even absent in corporate networks.

A major challenge, neglected so far by the research community, is MAC-layer greedy behavior: a station deliberately misuses the MAC protocol to gain bandwidth at the expense of other stations. The benefits of this misuse are the following.

- It can result in significant bandwidth gains as it directly deals with the wireless medium; therefore, it is more efficient than misbehavior at the network [11, 21] and transport [4] layers.
- It is hidden and independent from upper layers and hence cannot be detected by any mechanism designed for those layers. Thus, it can be combined with upper layer misbehavior to enhance it.
- It is always usable, since all the wireless stations use the same IEEE 802.11 MAC protocol; in contrast, for example, cheating with TCP [4] yields no benefits against UDP competing sources.

In this paper, we explore this space of MAC-layer greedy behavior. Rather than just presenting specific misbehavior techniques (as it is often the case in previous research), we propose a classification of the different MAC misbehavior techniques and illustrate them with representative examples, some of which are introduced for the first time. Then, we present DOMINO, a system for detecting MAC misbehavior

in a way that is transparent to the operation of the network. The key features of DOMINO are its (1) seamless integration in the AP[1] without interfering with its normal functions (this is achieved by means of a statistical passive approach based on traffic monitoring), (2) compatibility with existing networks, and (3) applicability to future versions of IEEE 802.11 with minor changes.

Based on the output of the detection system, the WISP (Wireless ISP) can decide how to react to cheating users. For example, the operator can charge a penalty bill, reduce the service quality, or even completely stop the service, depending on the extent of the observed cheating and the responsiveness of the cheater.

A major contribution of this work is that it goes beyond the theoretical consideration of the problem and presents the results of real experiments that demonstrate the ease of cheating and the efficiency of DOMINO. We succeeded, by means of minor changes to a driver for IEEE 802.11 compliant cards, in obtaining higher throughput at the expense of stations equipped with unmodified drivers. We also show how monitoring frames on the wireless medium enables the detection of greedy behavior.

This work is part of the Terminodes Project [15].

The rest of the paper is organized as follows: Section 2 overviews related work; Section 3 describes the system model; Section 4 explores the misbehaving techniques; Section 5 compares between different misbehavior metrics; Section 6 presents the detection system; Section 7 studies the proposed approach using simulations; Section 8 describes our implementation of the cheating and detection mechanisms; Section 9 discusses some relevant issues, and Section 10 concludes the paper.

## 2. RELATED WORK

The problem of MAC layer misbehavior is relatively new and unexplored in the literature. Kyasanur and Vaidya [20] have addressed the MAC layer misbehavior using detection and correction mechanisms; their paper was an important source of inspiration for our work. Their main idea is to let the receiver assign and send backoff values to the sender in CTS and ACK frames and then use them to detect potential misbehavior. The latter is handled using a correction scheme that adds to the next backoff a penalty that is a function of the observed misbehavior. This solution is efficient, but at the expense of the following issues.

- It requires a modification of the IEEE 802.11 MAC protocol in a way that is incompatible with the current standard. Such an approach is practically unfeasible.

- It gives control to the receiver over the sender, by making the former assign backoff values to the latter in both the detection and the correction schemes. Hence the proposed approach opens the door to new misbehavior techniques, including misbehaving receivers and collusion between sender and receiver.

[1]The actual hotspot component in which the system has to be installed is the *hotspot controller*, which provides access control and can control several APs [12]; nevertheless we assume in the following, without loss of generality, that the hotspot controller is incorporated in a single AP and thus we refer, for simplicity, to both components as AP.

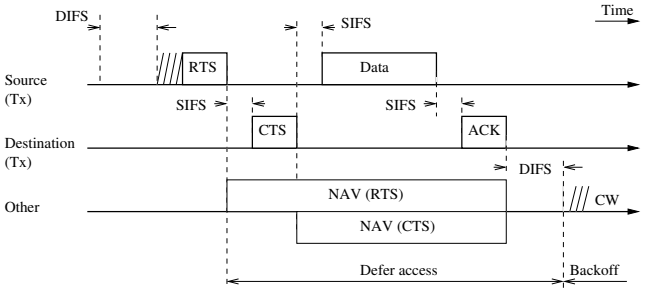



**Figure 1: The distributed coordination function (DCF) of IEEE 802.11 operating in RTS/CTS mode.**

- It creates communication and computation overhead. The first is due to the addition of new frame header fields and the second to the detection and correction schemes that have to compute backoffs and, in some cases, penalties for each individual frame of the sending station (in the infrastructure case, all this load will be centralized at the AP).

- It considers only stations with backlogged UDP traffic to detect misbehavior. But if the misbehaving station generates traffic with an interframe delay, the latter may result in the measured backoff being larger than the assigned one and hence leave the cheater undetected; this problem will be addressed in later sections.

Konorski [18] considers an ad hoc network in which all stations hear each other and he proposes a misbehavior-resilient backoff algorithm based on game theory. As it requires a new backoff mechanism, different from the current standard, this solution is not practical for current hotspots.

It is worth noting that none of the previous two works include a real implementation of the proposed algorithms.

Intrusion detection systems [27] are also relevant to the MAC layer, although they handle security flaws rather than protocol misbehavior. A commercial example of these systems is AirDefense Guard [2], in which distributed sensors, placed near APs, monitor the wireless medium and send reports to a central server. Our system can be installed on these sensors, hence integrating detection of intrusion and of misbehavior.

Various solutions to routing layer misbehavior in wireless ad hoc networks have been proposed in the literature (e.g., [9, 21]). However, the problem we consider in this paper focusing on the MAC layer, is too different to make those solutions eligible here.

## 3. SYSTEM MODEL

In the next sections, we use the following system model and assumptions.

- The IEEE 802.11 WLAN (AP and stations) works in the infrastructure mode using DCF (Distributed Coordination Function), which is the operation mode usually deployed.

  As shown in Fig. 1, DCF delays frame transmissions right after the channel is sensed idle for DIFS (DCF InterFrame Spacing) time. It waits for an additional ran-

dom time, *backoff time*, after which the frame is transmitted. The backoff time is bounded by the contention window size $CW$. This is applied to data frames in the basic scheme, and to RTS frames in the RTS/CTS scheme. The backoff time of each station is decreased as long as the channel is idle. When the channel is busy, the backoff time is frozen. When the backoff time reaches zero, the station transmits its frame. If the frame collides with another frame (or RTS), the sender times out waiting for the ACK (or the CTS) and computes a new random backoff time with a larger $CW$ to retransmit the frame with lower collision probability. When a frame is successfully transmitted, the $CW$ is reset to $CW_{min}$. The network allocation vector (NAV) of all other stations is set to the frame duration field value in RTS/CTS and DATA headers.

- We consider a single trusted AP operated by a WISP.
- Only user stations misbehave; if they do, they do so in a rational way, meaning that misbehavior is motivated by a beneficial outcome in terms of obtained throughput. Thus, we do not consider malicious misbehavior that aims at disrupting the functionality of the network.
- The detection system is implemented only at the AP. Thus, no modification nor reconfiguration of wireless adapters have to be made at the user side. In addition, the solution is under the full control of the AP and hence, of the WISP.

# 4. MISBEHAVIOR TECHNIQUES

In this section we present a taxonomy of MAC layer misbehaviors, introducing several new techniques that do not rely on security weaknesses of the standard and are simpler and more efficient than known methods. We can divide the MAC misbehavior space into two major dimensions as follows.

## 4.1 MAC greedy behavior

This category of misbehavior relies on modifying the operation of the IEEE 802.11 protocol by failing to follow communication procedures or changing parameters defined in the standard. An example of each modification follows, respectively.

- Selectively scramble frames sent by other stations in order to increase their contention windows. The frames to be targeted can be the following:
  1. CTS frames. In this case the cheater hears an RTS frame destined to another station and intentionally causes collision and loss of the corresponding CTS frame in order to prevent the subsequent long frame exchange sequence (RTS/CTS handshake is used for large frames). As a result, the channel becomes idle after the corrupted CTS and the cheater gets a chance to send its data.
  2. ACK and DATA frames. Although this does not result in saving the data frame transmission time, it causes the contention window of the ACK destination (i.e., the DATA source) station to be doubled and consequently makes the latter select larger backoffs. As before, the cheater increases its chances to get access to the channel.
- Manipulate protocol parameters to increase bandwidth share:
  1. When the channel is idle, transmit after SIFS but before DIFS.
  2. When sending RTS or DATA frames, increase the included NAV value in order to prevent the stations in range from contending during this time. A DoS attack using the same principle was described and evaluated in [8].
  3. Reduce the backoff time. This can be done by choosing a small fixed contention window; thus, the backoff is always chosen from this small window.

  A cheater may also combine several of the above techniques or adaptively change its misbehavior to avoid being detected. We will address this type of cheating in Section 9.3.

## 4.2 Security attacks

This category of attacks (e.g., the deauthentication attack [12]) exploits security weaknesses of the MAC protocol (such as flaws in authentication or encryption mechanisms) and targets the access control, confidentiality, or availability of the network. They may be rational or malicious (as defined in Section 3). An overview of these attacks can be found in [12]. As this category has been extensively addressed before, we will not consider it further in this paper.

In the rest of the paper, "misbehavior" means greedy behavior of stations and does not relate to the security aspects of wireless networks.

# 5. MISBEHAVIOR METRICS

In order to detect misbehavior while reducing false positives, the AP has to gather enough statistical data and then make decisions based notably on average values. Therefore, it needs to measure one or more attributes of the transmitting stations. In this section, we identify two such attributes, namely throughput and backoff, and discuss the pros and cons of each of them in order to choose the most suitable one for the detection system (presented in the next section).

## 5.1 Throughput

Although throughput seems to be the most intuitive metric for distinguishing stations using higher shares of the channel bandwidth than other stations, this metric would face several obstacles if used for detection.

If two stations have different data rates and delays, such as VoIP versus streaming video sources, the throughput of the latter will be naturally much larger than that of VoIP. Hence, we cannot rely on throughput without knowing the application running on each station (this would require each station to declare its currently communicating applications and the AP, which frequently implements only the physical and MAC layers, to analyze information originating from the application layers of the stations).

Experimental studies in [5] and [26] have shown that the throughput of a UDP source in a wireless network is affected by many factors, such as overhead, SNR, network and host

hardware, device drivers, and network protocol implementations in the operating system. The authors of [14] prove that the decrease of the bit rate of a single station (due to a bad channel) decreases the bit rates of all the other stations to values below that of the disadvantaged station. The negative effect of SNR on channel capture is explored in [25] (according to the authors, the results obtained in the infrastructure mode are identical to those observed in the ad hoc mode). All these factors lead to high differences in throughput even among stations sending at equal rates.

The performance of TCP over wireless networks has been studied experimentally in [26]. The authors explain that TCP coupled with the IEEE 802.11 MAC protocol result in performance degradation. Among the factors that contribute to the degradation are the congestion window, recovery mechanism, packet size, and timeout values of TCP as well as the acknowledgements, retransmission retry limit, and backoff mechanism of IEEE 802.11 MAC.

Hence, although the fairness of wireless networks has been evaluated [7, 17, 22] typically using Jain's fairness index [16] (which in turn uses channel bandwidth shares), throughput is far from being the optimal misbehavior metric, in our case.

## 5.2 Backoff

As we aim notably at detecting backoff manipulation, backoff measurement is the most direct way to detect cheaters (the next section explains how the AP estimates the backoff chosen by a station by monitoring the channel idle time). It is less dependent than throughput on various factors, some of which have been discussed before. Nevertheless, it still has its own problems, discussed hereafter.

1. From the receiver point of view, a sender's backoff is the idle period that starts at the end of a DIFS after the last transmission until the next transmission from the sender, not including the transmission cycles (e.g., DATA-SIFS-ACK-DIFS) of other stations. This value is undistinguishable from the delay of a low packet rate source. This source can hence cheat while giving the impression of a well-behaved station. Even if the cheater increases its packet rate later, the previously collected large delays would increase its observed average backoff, thus decreasing the responsiveness of the system during a monitoring period (defined in the next section).

2. The MAC header does not include any information that can be used to compute the sender's backoff at the receiver. This leads to confusing conclusions after a collision, since the stations involved in the collision double their contention window and choose a new backoff value, whereas the other stations do not.

3. If the receiver senses the channel busy while the sender senses it idle, the latter will transmit after a short backoff from the receiver's point of view. This may be caused for example by the hidden terminal problem.

The shortcomings of the backoff detection metric can be more easily overcome than those related to throughput. In order to show this, we have devised several different backoff tests (discussed in the next section) that are complementary in the sense that none of them is enough to make a complete solution, but together they achieve the required detection ability.

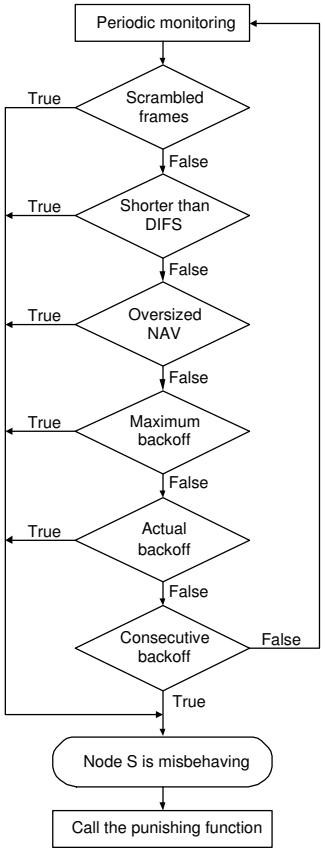



**Figure 2: Components of DOMINO. Each test manages its local counter of cheating; if this counter exceeds the corresponding threshold, the remaining tests are aborted to save resources, and control is transferred to the punishing mechanism. Otherwise, the tests are repeated after the next monitoring period.**

# 6. COMPONENTS OF DOMINO

In this section, we present the way to detect the misbehavior techniques described in Section 4.1. The complete detection system is depicted in Fig. 2. As mentioned before, this system has to be implemented only at the AP. The details of the system components are explained below.

Traffic traces of sending stations are collected periodically during short intervals of time called *monitoring periods* (the choice of their length is discussed in Section 9.4). The gathered data are then passed to several tests within the encapsulating DOMINO algorithm:

```
loop
  if monitoring period elapsed since last check then
    for each active station S_i do
      for j = 1 to 6 do
        execute Test j
```

The tests described below make use of the following function where $x$ indicates the test number.

**check**$_x(S_i, condition_x)$:

```
if condition_x is true then
  cheat_count_x(S_i) := cheat_count_x(S_i) + 1
  if cheat_count_x(S_i) > K_x then
    S_i is misbehaving
    call the punishing function
else if cheat_count_x(S_i) > 0 then
  cheat_count_x(S_i) := cheat_count_x(S_i) − 1
```

To decrease the number of false positives, a station should be suspected at least $K_x$ times (i.e., after at least $K_x$ monitoring periods, as defined in Fig. 2) before being considered a cheater. In addition, each time a station does not cheat, its $cheat\_count_x$ is decremented (until it reaches zero) to reward the correct behavior; this adaptivity also reduces the effect of erroneous detection of well-behaved stations. Although $K_x$ slightly reduces the responsiveness of the system, it can be small enough to prevent temporal but beneficial, i.e., long enough misbehavior from escaping detection.

It should be noted that all the tests described below are performed on each data sample successfully collected for a station $S_i$ during the last monitoring period; if misbehavior is detected, the checking on $S_i$ is interrupted as no further analysis is needed. For clarity, we present the operation of the tests on a single data sample.

## 6.1 "Scrambled frames"

In order to gain a significant share of the common wireless bandwidth using CTS/ACK/DATA scrambling, the cheater has to scramble a relatively large percentage of CTS, ACK, or DATA frames sent by other stations. As a result, its average number of retransmissions will be less than that of other stations, and it can be detected using Test 1. $num\_rtx(S)$ is the number of times station $S$ retransmitted its last frame successfully received by the AP; $\phi$ is a tolerance parameter with a value between 0 and 1.

**Test 1** Scrambled frames

$condition_1 := num\_rtx(S_i) < \phi \times E_{j \neq i}[num\_rtx(S_j)]$
call **check**$_1(S_i, condition_1)$

The punishing function depends on the policy of the WISP as mentioned in Section 1.

The system can detect a retransmission by observing a repeated sequence number in the header of RTS or DATA frames when the corresponding CTS or ACK frames are scrambled, respectively. In the case of DATA frames, one might argue that the AP would not be able to distinguish retransmissions because the DATA frames are scrambled. However, the cheater cannot scramble the headers of these frames, otherwise it cannot know if the frame is destined to it. Hence the system can still detect retransmissions by observing repeated sequence numbers in the MAC headers.

## 6.2 Detection of manipulated protocol parameters

In the following paragraphs we address misbehavior techniques that alter protocol parameters. We focus mainly on backoff manipulation since it is the easiest to implement (as we will show in Section 8) and the hardest to detect.

**"Shorter than DIFS"**

The AP can monitor the idle period after the last ACK and distinguish any station that transmits before the required DIFS period. After having observed this misbehavior repeatedly for several frames from the same station, the AP can make a reliable decision (Test 2).

**Test 2** Shorter than DIFS

$condition_2 := idle\_time\_after\_ACK(S_i) < DIFS$
call **check**$_2(S_i, condition_2)$

**"Oversized NAV"**

By measuring the actual duration of a transmission (including the DATA, ACK, and optional RTS/CTS) and comparing it with the NAV value in the RTS or DATA frame headers, the AP can detect stations that regularly set the NAV to very large values. In Test 3, the tolerance parameter $A$ (greater than 1) ensures that the AP does not mistakenly catch well-behaved stations.

**Test 3** Oversized NAV

$condition_3 := NAV(S_i) > A \times tx\_duration(S_i)$
call **check**$_3(S_i, condition_3)$

**Backoff manipulation**

*"Maximum backoff"*

Since the IEEE 802.11 protocol selects backoffs randomly from the range $[0, CW - 1]$ (where $CW$ depends on the number of retransmissions), the maximum selected backoff over a set of frames sent by a given station (without interleaving collisions; otherwise the contention window will be doubled) should be close to $CW_{min} - 1$, if the number of samples is large enough. The *maximum backoff* test (Test 4) uses this property to suspect stations whose maximum backoff over a set of samples is smaller than a threshold value $threshold_{maxbkf}$. Clearly, a tradeoff exists between the number of samples and the threshold; if we increase the threshold (its largest value is $CW_{min}$), we have to increase the number of sampled backoffs to get more distinct values and thus avoid false positives. In our simulations (Section 7), we use a threshold equal to $CW_{min}/2$; thus, the test works if the reduced contention window is in $[0, CW_{min}/2 - 1]$.

**Test 4** Maximum backoff

$condition_4 := max\_bkf(S_i) < threshold_{maxbkf}$
call **check**$_4(S_i, condition_4)$

Unfortunately, this test may be easily tricked by a clever cheater that succeeds at making the monitor observe in every sample at least one backoff value larger than or equal to the threshold; channel conditions can also yield a similar result and thus make the test fail. Thus, the *maximum backoff* test is only auxiliary to the next tests.

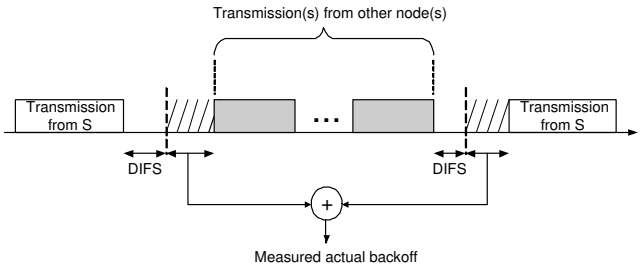



**Figure 3: Measurement of the *actual backoff*. Transmissions from $S$ are interleaved with one or more transmissions from other nodes (including the AP). The transmission includes in addition to the DATA frame all the control frames, such as RTS, CTS, and ACK, as well as the interleaving idle periods of SIFS. The measured value is the sum of all idle intervals (not including interframe spaces) between two transmissions from $S$.**

*"Actual backoff"*

This test (Test 5) consists in measuring the actual backoff as shown in Fig. 3. The main procedures of the test can be summarized as follows:

- If between two transmissions from a station $S$ there are no collisions, we assume that $S$ spent all its idle time backing off (although it may be just part of the $S$'s interframe delay). Then we estimate this backoff by computing the sum as illustrated in Fig. 3.
- If a collision happens, it is not possible to know the identities of the senders of the colliding frames and hence the stations whose measured *actual backoff* should be updated. To avoid complexity, collisions are simply not taken into account and both the current backoff and the next one are not measured for any station.[2]

**Test 5** Actual backoff

$condition_5 := B_{ac}[S_i] < \alpha_{ac} \times B_{acnom}$
call $\mathbf{check}_5(S_i, condition_5)$

In Test 5, $B_{ac}[S_i]$ denotes the average *actual backoff* (observed by the AP) of station $S_i$. $B_{acnom}$ is the nominal backoff value, which is equal to the average backoff of the AP if it has enough traffic to compute this value (according to usage studies in [19] and [24], the inbound traffic from the AP is usually larger than the outbound traffic). If the AP does not have enough data to derive a nominal backoff value from its own traffic, it uses an analytical value $E[B_{ac}]$ (derived in the Appendix). We do not use the analytical value in the first place since it depends on the number of active stations and is computed assuming backlogged sources. In a practical setting, this assumption might be wrong due to mobility and usage patterns (Tang and Baker [24] find that, 80% of the time, peak throughput is due to a single user and application, which is typically a large file transfer).

[2]Stations that hear frame headers with wrong CRC, caused by a collision, will defer their transmissions by EIFS (Extended InterFrame Spacing). This latter does not interfere with the measurements since all deferrals of all nodes are not taken into account after a collision.

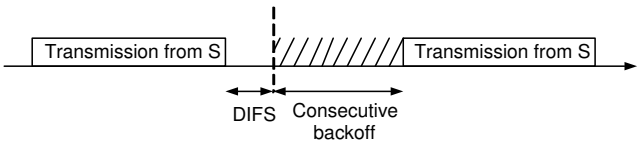



**Figure 4: Measurement of the *consecutive backoff*. Backoff values are taken only between consecutive non-interleaved transmissions from $S$.**

The $\alpha_{ac}$ ($0 < \alpha_{ac} \leq 1$) parameter is configurable according to the desired true positive (correct detection) and false positive (wrong detection) percentages (e.g., we use $\alpha_{ac} = 90\%$ in our simulations). To reduce false positives, we use $K_5 = 3$ (defined in the function $check_x$) in our simulations; this shows that this value can be small enough to quickly detect cheaters without accusing well-behaved stations.

As it collects no data during collisions, the actual backoff test measures backoffs that are selected only from the $[0, CW_{min} - 1]$ range. Due to its mechanism, this test fails to detect the misbehavior case when the cheater has interframe delays (e.g., a TCP source using congestion control). In fact, the test measures these delays instead of backoffs because it adds up the idle periods between transmissions from the same source (Fig. 3). Hence, although the chosen backoffs may be subject to cheating, the monitor will not be able to measure them correctly; the solution to this problem is provided by the *consecutive backoff* test.

*"Consecutive backoff"*

Fig. 4 illustrates this test (Test 6), which works in the case of sources with interframe delays. In practice, this is mainly the case of TCP sources (in this case the delay is due to the congestion control of TCP), which represent over 91% of traffic in real networks [6, 19]. The *actual backoff* test for these sources does not yield the correct values (as explained in the previous paragraph), and consequently cannot detect potential cheating.

Let us consider a station $S$ sending TCP traffic and being monitored by DOMINO. We assume that there is enough traffic from other sources on the common channel such that, between two frames sent by $S$ and separated by a transport layer delay, there is at least one interleaving frame from another station. Hence, if the AP observes two consecutive non-interleaved frames from $S$, it can consider the idle time between them as only a backoff in addition to the mandatory DIFS. These consecutive frames are the result of channel contention that may force $S$ to queue packets at the MAC layer even if they were separated by a delay at upper layers. In this situation, $S$ would benefit from cheating with backoff in order to free its MAC layer queue. Thus, the system can collect significant samples of the backoff values chosen by $S$; we call these samples *consecutive backoffs*.

The above assumption of traffic level is realistic. In fact, if the traffic on the channel is low enough to invalidate this assumption, i.e., if $S$ can send consecutive non-interleaved frames separated by a delay in addition to the backoff and

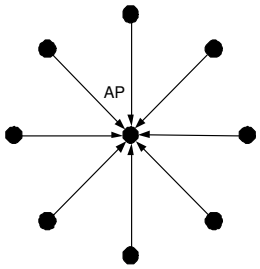



**Figure 5: Simulation scenarios: 8 stations send UDP or TCP data to the AP, which also generates traffic similarly to one station. The distance between the stations and the AP is 50m. All stations are within range of each other.**

DIFS, cheating would be pointless since reducing the backoff does not affect the upper layer delay. Misbehavior detection would not be needed in this case.

**Test 6** Consecutive backoff

$condition_6 := B_{co}[S_i] < \alpha_{co} \times B_{conom}$

call $\mathbf{check}_6(S_i, condition_6)$

As with the previous test, the average of the collected values $B_{co}[S_i]$ is compared to a fraction $\alpha_{co}$ (we use $\alpha_{co}$ = 90% in our simulations) of the nominal value $B_{conom}$. The latter is the average consecutive backoff of the AP if enough data are available. Otherwise, it is an analytical value $E[B_{co}]$ (computed in the Appendix). As in the *actual backoff* test, we use $K_6 = 3$ in our simulations to decrease the number of false positives.

# 7. SIMULATION RESULTS

In order to study the performance of the proposed solution, we have used ns-2 with the Monarch project extension [13] to simulate our detection system. As the frame scrambling misbehavior is fairly easy to detect using the number of retransmissions, this section examines in detail only the backoff manipulation tests and the complete detection mechanism. Although these tests are capable of detecting multiple cheaters, in the simulations we have focused on the case of a single cheater to simplify the presentation of the results.

## 7.1 Simulation topology

Following the discussion in the previous section about the effect of traffic on Tests 5 and 6, we will study two cases (Fig. 5). Due to the lack of space, we cover only these scenarios that represent the most common traffic types.

1. UDP traffic

   Besides the cheater, there are 7 stations sending CBR traffic (the nominal rate is 500 bytes/packet, 200 packets/s); the cheater is also a CBR source.

   In any idle slot, there is at least one packet ready for transmission by any of the competing stations. The time elapsed between two transmissions from the same station (interleaved with transmissions from other stations) is therefore due only to the backoff chosen by the IEEE 802.11 protocol.

2. TCP traffic

   Each of the 8 stations runs an FTP application; one station is cheating.

   This case illustrates the effect of interframe delays (due to TCP congestion control) on backoff measurement. This is the most realistic scenario.

In both cases the AP generates traffic similarly to one station, i.e., CBR in the first case and FTP in the second.

To take into account the fading effects present in real channels, we have used the shadowing channel model that is represented by the following equation:

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

where $P_r(d)$ is the mean received power at distance $d$, $d_0$ is a reference distance, $\beta$ is the path loss exponent, and $X_{dB}$ is a Gaussian random variable with zero mean and standard deviation $\sigma_{dB}$. We have used $\beta = 2$ (free space propagation) and $\sigma_{dB} = 4$.

Results are averaged over 10 simulations, 110s each. The monitoring period is set to 10s, which also corresponds to one decision (cheater or well-behaved) by the AP regarding each station. Thus, each point on the following graphs is averaged over 100 samples with a 95% confidence interval; the first 10s of each simulation is an initialization period.

In the following, the *misbehavior coefficient* represents the amount of misbehavior. A *misbehavior coefficient* equal to $m$ means that the corresponding station uses a fixed contention window equal to $(1-m) \times CW_{min}$ and then chooses its backoff from this new window. Thus, $m = 0$ means no misbehavior, and $m = 1$ means that the station transmits without any backoff.

## 7.2 Impact of misbehavior on throughput

Before presenting the performance of the detection system, we compare the throughput values of cheating and well-behaved stations in both simulation scenarios.

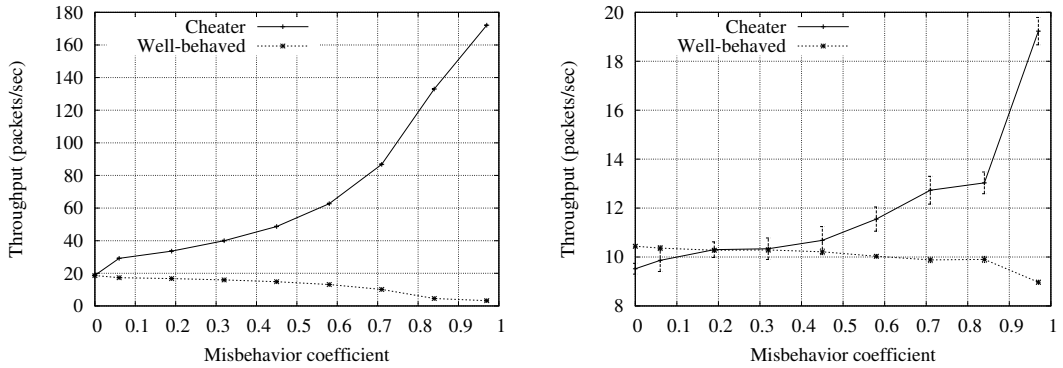
Fig. 6 shows that MAC misbehavior results in throughput[3] benefits that are obtained at the expense of well-behaved stations and that increase with the amount of misbehavior. We can also notice that this increase is less significant in the case of TCP sources. This is due to the TCP congestion control mechanisms and the dependence of the TCP throughput, including the cheater's, on the rate of the TCP ACKs, which are sent by the (well-behaved) AP.

## 7.3 Actual backoff

From the simulation graphs we can draw the following observations:

- In the UDP traffic case, the test performs well, as shown in Fig. 7, because there is always at least one frame ready for transmission by each station. Hence the channel idle time between two transmissions from a station is the result of only the backoff mechanism (in addition to the DIFS).

[3]The graphs also display confidence intervals, which are very small in some cases.




(a) UDP traffic

(b) TCP traffic

**Figure 6: Throughput comparison between misbehaving and well-behaved stations.**

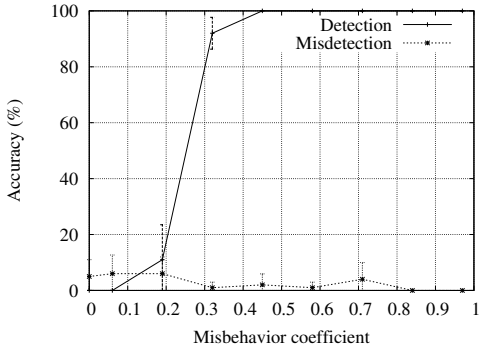



**Figure 7: *Actual backoff* test in the UDP traffic case.**

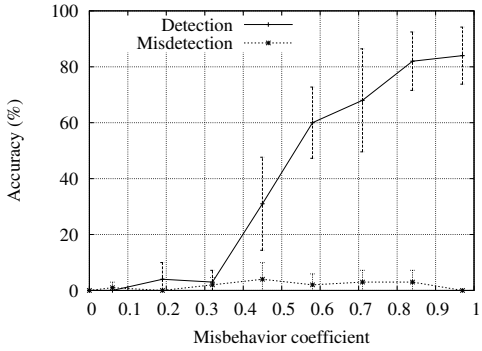



**Figure 8: *Consecutive backoff* test in the TCP traffic case.**

- In the TCP traffic case, the numbers of both correct and wrong detections are very small (the curves are practically superimposed with the x-axis; thus, the corresponding figure does not provide any important information and hence we omit it). The low correct-detection accuracy can be explained by the fact that the measured actual backoff is actually the idle period (not including transmission cycles) between two interleaved transmissions from the same station, which is equal in this case to the delay between frame transmissions from the source. This delay is created by the TCP congestion control mechanisms.

## 7.4 Consecutive backoff

The performance of this test differs from that of the previous one for the reasons mentioned in the description of the test (Section 6.2) and confirmed by simulations.

In the UDP traffic case, the results of the test are of no use (the curves are superimposed with the x-axis and therefore we omit them). The reason is that in this case the measured average consecutive backoff rapidly decreases with the number of stations (as the Appendix shows, the analytical average value steeply decreases with the number of stations). The comparison of small values becomes inaccurate, thus seriously affecting the test significance.

In the TCP traffic case, the test yields good results as Fig. 8 shows. This is due to the presence of other sources that do not allow the source with the interframe delay (induced by congestion control) to transmit two frames consecutively without having queued the second one, i.e., the delay does not affect the idle time between two consecutive non-interleaved transmissions from the source. Otherwise, if there is no frame ready in the queue, another source takes control over the channel and transmits at least one frame between two successive frames of the first source.

## 7.5 Complete mechanism

The descriptions of the *actual backoff test* and the *consecutive backoff test* in Section 6.2, as well as the simulation graphs presented so far, have shown that each test performs well in specific traffic scenarios. The complete mechanism is thus a combination of both.

- The *actual backoff test* catches misbehavior with high accuracy in the UDP traffic case and yields no detection nor misdetection in the TCP traffic case.

- On the contrary, the *consecutive backoff test* detects misbehavior in the TCP traffic case but does not work in the UDP traffic case.
- Hence the full scheme uses both tests in order to detect misbehavior in all traffic scenarios while keeping the misdetection percentage very low.

It is worth noting that, as long as there is enough traffic on the channel to satisfy the assumption in Test 6, only the type of the sender traffic determines which test works and hence misbehavior in mixed-traffic scenarios (TCP/high rate UDP) can also be accurately detected. If the traffic on the channel is low, misbehavior does not yield substantial throughput benefits hence its detection is not necessary.

# 8. IMPLEMENTATION

To prove the need for and the efficiency of the proposed detection system, we have implemented a prototype misbehavior scenario, including a prototype of DOMINO, which we describe in this section.

## 8.1 Experimental setup and equipment

As Fig. 9 illustrates, the prototype consists of two senders (user stations), one of which is cheating, a receiver (the AP), and a monitor station. All 4 stations are laptops equipped with Proxim's ORiNOCO 11a/b/g Gold Combo-Card wireless cards based on the Atheros AR5212 chipsets. The MADWIFI (Multiband Atheros Driver for WiFi) driver [1] provides full support for wireless adapters using Atheros chipsets on Linux platforms (kernel 2.4.20 or higher; we used version 2.4.20). It can be used as a loadable kernel module, thus avoiding the recompilation of the kernel after making changes to the driver. The numerous features of this driver make the implementation of cheating and monitoring easy, as we describe next. Although the same functionality may not be available in products from other manufacturers, the rising trend of shifting more firmware and hardware functions to the software side will make cheating easily implementable on most chipsets.

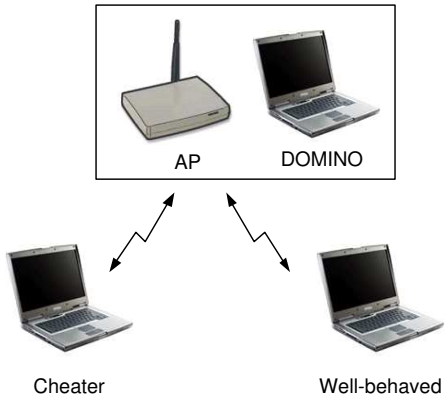



**Figure 9: Experimental setup.**

## 8.2 Misbehavior

To illustrate the feasibility of cheating with almost no overhead, we have implemented one of the cheating methods described in Section 4; specifically, we have modified the value of the contention window used in the backoff procedure.

As $CW_{min}$ and $CW_{max}$ values are stored in the same register[4], all that has to be done to enable cheating is to write to this register the desired $CW$ values in the driver code. We use the C function:

```
writel(value, register_address);
```

in the device *initialization*, *reset*, *mode change* (between .a, .b, and .g modes), and *scan* functions since the register values are reset in each of them.

## 8.3 Monitoring

The MADWIFI driver has a built-in `MONITOR` mode that allows for the capturing of IEEE 802.11 beacon and control frames. In addition, the driver adds a *prism2* header to the frames in order to support sniffing applications, such as Ethereal and Kismet, that use the *libpcap* capture file format and can display the captured frames. Fig. 10 shows a snapshot of wireless data collected in `MONITOR` mode and displayed by Ethereal. Since a wireless card using the MADWIFI driver can be in `MONITOR` mode exclusively, we use an additional laptop to play the role of the monitor. Although the monitoring function can also be implemented on the AP itself, we have not addressed this issue as the driver is still under development.

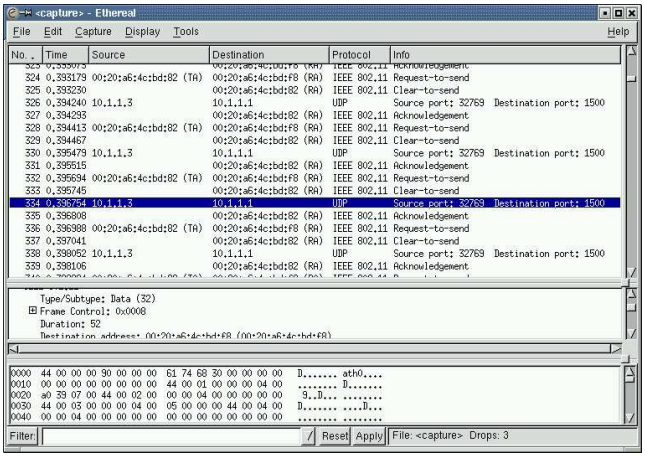
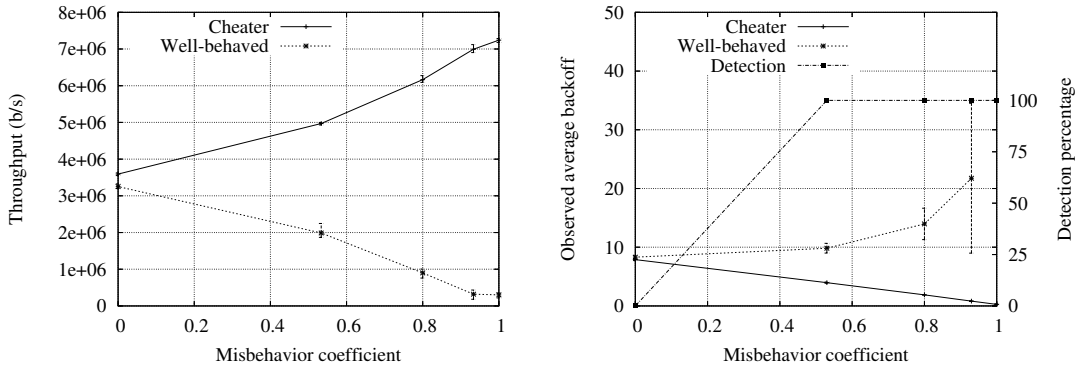



**Figure 10: Snapshot of Ethereal displaying captured wireless frames. The timestamps in column 2 can be used to calculate backoff values as explained in Section 8.3.**

To observe the behavior of other stations, the monitoring station should be placed near the AP, on which the monitoring mechanism should be typically installed. We use Ethereal to display the captured data and `awk` scripts to analyze the corresponding trace files. Specifically, each frame is associated with a timestamp indicating when its reception was completed. As described in Section 6, we measure backoff values from the time elapsed between the end of an ACK and the beginning of the next RTS or DATA frame; therefore, we need to know when the reception of an RTS

[4]For deontological reasons, we refrain from publishing the value of the register address. However, it is very easy to obtain.




(a) Throughput (b) Backoff

**Figure 11: Experimental results. The values correspond to the *misbehavior coefficients* 0, 0.53, 0.8, 0.93, and 1 as explained in Section 8.4.**

or DATA frame starts. Since the AP knows the payload size (after the whole frame has been received) and the used bit rate, it can compute the transmission time of the DATA frame and hence the beginning of this transmission. The DIFS value being defined by the standard (e.g., it is $50\mu s$ in IEEE 802.11b), the AP can compute the backoff values, as Figures 3 and 4 illustrate.

## 8.4 Results

We have performed experiments corresponding to several values of the cheater contention window, which should be of the form `(power of 2) - 1`. Specifically, we have set both $CW_{min}$ and $CW_{max}$ to 0, 1, 3, 7, 15 (the default value of $CW_{min}$ is 15 and that of $CW_{max}$ is 1023 on the wireless cards that we have used), which correspond to *misbehavior coefficients* of 1, 0.93, 0.8, 0.53, and 0, respectively. We have observed the resulting throughput (Fig. 11(a)) and backoff (Fig. 11(b)) of the cheating and well-behaved stations.

In Fig. 11(a) we can see that the cheater obtains higher throughput, at the expense of the well-behaved station, by increasing its misbehavior. The corresponding observed backoff values are shown in Fig. 11(b) along with the detection curve. When the misbehavior percentage increases, the cheater's average backoff decreases (thus increasing its chances to grab the channel first and boosting its throughput). This can be easily detected by our mechanisms as the detection curve shows. In the meantime, the average backoff of the well-behaved station increases with the misbehavior percentage (due to collisions and the following increase of the contention window); this explains its decreasing throughput.

## 8.5 From prototype to product

When transforming the prototype into a real product, several implementation choices can be distinguished:

- DOMINO can be fully implemented in software (or firmware, depending on the AP). The system, which can run as a module of the AP software, will collect information about frames, such as arrival time and size, received and sent by the AP and periodically analyze the collected data. We consider exploring this approach once the MADWIFI driver reaches the production stage.

  The main disadvantage of a software implementation is that upgrade and maintenance operations might create time overhead by temporarily interrupting the service offered by the AP.

- An alternative would be to install DOMINO on a wireless card separate from the AP, but physically close to it in order to view the network from the AP perspective as much as possible. This approach is similar to the AirDefense Guard intrusion detection system mentioned in Section 2.

  While removing the time overhead and potential service interruption associated with a pure software solution, the hardware option creates space and cost overhead due to the additional equipment required to operate the system.

The choice of the implementation option will depend on several factors, such as service requirements (the downtime of some heavily used hotspots may incur higher losses than the cost of deployment of a hardware-implemented DOMINO system), available equipment (it may be easier to deploy hardware DOMINO sensors than modifying the firmware of some existing APs) and infrastructure (hardware sensors will require additional space and cabling). Thus, the WISP and DOMINO system provider have to evaluate all the important factors in a particular network before choosing one of the above options.

# 9. DISCUSSION

This section addresses some additional issues related to the detection system.

## 9.1 Hidden terminals

Hidden terminals may have a negative effect on the detection system. For example, if two stations A and B are seen by the AP but hidden from each other, A may sense the medium idle while the AP senses it busy because B

transmits. As a result, A will keep decrementing its backoff counter and then transmit a frame whose backoff measured at the AP will appear smaller than the actual value. After several repetitions of this scenario, the detection mechanism will output a wrong suspicion of A. By choosing appropriate values for detection thresholds (specifically, $\alpha_{ac}$, $\alpha_{co}$, $K_5$, $K_6$ defined in Section 6.2) in both simulation scenarios, i.e., by tolerating some misbehavior, we have managed to reduce false positives in the presence of hidden terminals.

## 9.2 Security

It should be noted that DOMINO can be exploited to create *hybrid* attacks, taking advantage of both security flaws and MAC vulnerability. For example, a cheater may impersonate a well-behaved station to provoke its punishment and, possibly, its disconnection from the network by the operator. But a deauthentication attack [12], which is easier to perpetrate, would yield a similar effect without relying on the punishment policy. In addition, the adoption of new security mechanisms, such as WPA (Wi-Fi Protected Access) and IEEE 802.11i [12], would limit the efficiency of these hybrid attacks. In fact, the cheater cannot transfer useful data in the faked frames because it does not know the encryption key of the impersonated host. As a result, such an attack would incur on the cheater an overhead due to the dummy frames it sends. The solution to these attacks lies in the use of the enhanced security mechanisms jointly with DOMINO. We will consider the details of this solution in our future work.

## 9.3 Adaptive cheating

We call *adaptive cheating* the set of misbehavior techniques that exploit some knowledge about the way DOMINO works. For example, a cheater may switch frequently enough between several techniques described in Section 4.1, in such a way that the system fails to collect enough data to detect misbehavior. But as the cheater does not know the detection parameters, such as the monitoring period and the thresholds, it will be hard to adapt to the detection system in order to avoid being caught.

Another way of tricking DOMINO would consist in employing techniques to disable some tests. For example, a cheater might intentionally cause collisions between two of its frames to fail the *actual backoff* test and never transmit two consecutive non-interleaved frames to fail the *consecutive backoff* test. But such techniques obviously increase the cheater's overhead (e.g., in terms of interframe delay) that might not be compensated by a compelling throughput advantage over other stations.

## 9.4 Monitoring period

To avoid overloading the AP with per-frame computations, the data required for detection are collected during configurable intervals of time; at the end of each interval, the detection mechanism is run. Another advantage of this method over a per-frame detection approach is the ability to collect more statistical data and hence increase the accuracy. In addition, it has been shown [7, 17, 22] that the binary exponential backoff algorithm of IEEE 802.11 is unfair in the short term. This would result in false positives if stations were monitored over short term periods even in the absence of misbehavior. Therefore the monitoring period has to be large enough to achieve long term backoff fairness.

Taking into account the typical bit rates, monitoring periods can be short enough (as was shown in the simulations) to prevent the cheater from gaining large benefits before being detected. For example, assuming 500 byte packets and 7Mbps data rate (this is the maximum effective IEEE 802.11b rate) equally divided among 50 stations, the AP can collect in 10 seconds 350 backoff values per station.

## 9.5 Implementation overhead

As DOMINO is passive, there is no communication overhead in our solution. In addition, the required storage and computation overhead is very small. As an example, we have calculated rough upper bounds of the overhead of the backoff tests. We need to record the average backoff values (specifically, the *maximum, consecutive* and *actual* backoffs) over the monitoring period for each transmitting station; thus, the required storage is roughly proportional to the number of transmitting stations.

For example, the AP needs 8 integer registers per station to store the aggregated statistical values; thus the required storage for 50 stations is a mere 1600 bytes (assuming 4 byte registers). The processing of each frame received by the AP requires at most 6 arithmetic instructions; if we assume that the channel is used at full bandwidth (which can be translated to approximately 7Mbps effective data rate for IEEE 802.11b WLANs) and the packet size is 500 bytes, the computation overhead is less than 10500 arithmetic instructions or 0.021% of the CPU time (if we assume that an instruction takes 4 CPU clock cycles and the processor speed is 200MHz). Hence, a typical AP with a 200MHz processor and 16MB RAM can run the detection system without a noticeable performance difference.

# 10. CONCLUSION AND FUTURE WORK

MAC layer misbehavior in IEEE 802.11 networks can lead to severe unfairness in bandwidth distribution. This can become a serious problem in public Internet access hotspots where individual users have to pay for network usage and hence may be motivated to cheat in order to increase their share of the medium.

In spite of its relevance, this topic is still relatively unexplored in the research community. In this paper, we have classified MAC layer misbehaviors, presented some new techniques, and provided the corresponding detection mechanisms. In contrast with previous papers that have proposed modifications to the MAC protocol, thus requiring a modification of existing wireless cards, we have developed a solution that can be completely integrated in the AP and uses only statistical data analysis. An important feature is that a cheater has no way of knowing whether an AP is DOMINO-enabled.

Using simulations, we have shown that DOMINO achieves high accuracy of detection in a variety of scenarios. The system is resilient to several factors, such as traffic types, that can affect the performance of other detection techniques. Hence, the main features of the proposed solution are its efficiency and applicability to real networks.

Another important contribution of this paper is the cheating and detection prototype that we have implemented and that shows the ease of cheating, as well as the simplicity and efficiency of the proposed detection system.

We believe that the scope of this paper goes beyond IEEE 802.11 networks; indeed, we provide a framework that can

be adapted to the study of cheating and detection techniques in any network based on shared spectrum.

For future work, we consider addressing in more detail the case of adaptive cheating. We will also explore the effect of mobility on the system. In regard to implementation, we will introduce these and other enhanced features in the final version of the MADWIFI driver.

It should also be noted that in this paper we have focused on attacks aimed at the traffic outgoing from the stations. In our future efforts we will expand the set of attacks to new techniques that decrease the traffic incoming to the stations from the AP, thus favoring the cheater's incoming traffic.

## Acknowledgements

We would like to thank Mario Cagalj, Jean-Dominique Decotignie, Amre El-Hoiydi, Saurabh Ganeriwal, Matthias Grossglauser, Edward Knightly, Maciej Kurant, Jean-Yves Le Boudec, and the anonymous reviewers for their helpful feedback on earlier versions of this work. We would like also to thank Hari Balakrishnan for shepherding this paper.

## 11. REFERENCES

[1] `http://sourceforge.net/projects/madwifi`.

[2] `http://www.airdefense.net`.

[3] IEEE Standard for Wireless LAN-Medium Access Control and Physical Layer Specification, P802.11, 1999.

[4] A. Akella, S. Seshan, R. Karp, and S. Shenker. Selfish behavior and stability of the internet: A game-theoretic analysis of TCP. In *Proceedings of SIGCOMM'02*, 2002.

[5] M.G. Arranz, R. Aguero, L. Munoz, and P. Mahonen. Behavior of UDP-based applications over IEEE 802.11 wireless networks. In *Personal, Indoor and Mobile Radio Communications, 12th IEEE International Symposium on*, volume 2, pages F–72–F–77, Sep/Oct 2001.

[6] A. Balachandran, G. Voelker, P. Bahl, and P. Rangan. Characterizing user behavior and network performance in a public wireless LAN. In *Proceedings of ACM SIGMETRICS'02*. ACM Press, June 2002.

[7] C. Barrett, M. Marathe, D. Engelhart, and A. Sivasubramaniam. Analyzing the short-term fairness of IEEE 802.11 in wireless multi-hop radio networks. In *Modeling, Analysis and Simulation of Computer and Telecommunications Systems, 10th IEEE International Symposium on*, pages 137–144, 2002.

[8] J. Bellardo and S. Savage. 802.11 denial-of-service attacks: Real vulnerabilities and practical solutions. In *Proceedings of USENIX Security Symposium*, August 2003.

[9] N. Ben Salem, L. Buttyán, J.-P. Hubaux, and M. Jakobsson. A charging and rewarding scheme for packet forwarding in multi-hop cellular networks. In *Proceedings of MobiHOC'03*, 2003.

[10] G. Bianchi. Performance analysis of the IEEE 802.11 Distributed Coordination Function. *IEEE Journal on Selected Areas in Communications*, 18(3):535–547, March 2000.

[11] L. Buttyán and J.P. Hubaux. Stimulating cooperation in self-organizing mobile ad hoc networks. *ACM/Kluwer Mobile Networks and Applications*, 8(5), October 2003.

[12] J. Edney and W. A. Arbaugh. *Real 802.11 security: Wi-Fi Protected Access and 802.11i*. Addison-Wesley, 2004.

[13] K. Fall and K. Varadhan. *ns notes and documentation*. UC Berkeley, LBL, USC/ISI, Xerox PARC, 2003.

[14] M. Heusse, F. Rousseau, G. Berger-Sabbatel, and A. Duda. Performance anomaly of 802.11b. In *Proceedings of INFOCOM'03*. IEEE, 2003.

[15] J. P. Hubaux, Th. Gross, J. Y. Le Boudec, and M. Vetterli. Towards self-organizing mobile ad-hoc networks: the terminodes project. *IEEE Comm Mag*, 39(1):118 –124, January 2001.

[16] R. Jain. *The Art of Computer Systems Performance Analysis*. John Wiley and Sons, 1991.

[17] C. Koksal, H. Kassab, and H. Balakrishnan. An analysis of short-term fairness in wireless media access protocols. In *Proceedings of ACM SIGMETRICS'00*, June 2000.

[18] J. Konorski. Multiple access in ad hoc wireless LANs with noncooperative stations. In *NETWORKING*, volume 2345 of LNCS, Springer, 2002.

[19] D. Kotz and K. Essien. Analysis of a campus-wide wireless network. In *Proceedings of MOBICOM'02*, pages 107–118, September 2002.

[20] P. Kyasanur and N. Vaidya. Detection and handling of MAC layer misbehavior in wireless networks. In *Dependable Systems and Networks*, June 2003.

[21] S. Marti, T.J. Giuli, K. Lai, and M. Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *Proceedings of MOBICOM'00*, pages 255–265, 2000.

[22] T. Nandagopal, T. Kim, X. Gao, and V. Bharghavan. Achieving MAC layer fairness in wireless packet networks. In *Proceedings of MOBICOM'00*, pages 87–98, 2000.

[23] ABI Research. Wi-Fi hotspot equipment markets: Business case analysis through deployment and subscriber forecasts. `http://www.abiresearch.com/reports/WLPS.html`, 2003.

[24] D. Tang and M. Baker. Analysis of a local-area wireless network. In *Proceedings of MOBICOM'00*, pages 1–10. ACM Press, August 2000.

[25] C. Ware, J. Judge, J. Chicharo, and E. Dutkiewicz. Unfairness and capture behaviour in 802.11 ad hoc networks. *IEEE International Conference on Communications*, 1:159 –163, 2000.

[26] G. Xylomenos and G. Polyzos. TCP and UDP performance over a wireless LAN. In *Proceedings of INFOCOM'99*, volume 2, pages 439–46. IEEE, March 1999.

[27] Y. Zhang and W. Lee. Intrusion detection in wireless ad hoc networks. In *Proceedings of MOBICOM'00*, pages 275–283, 2000.

# APPENDIX

As mentioned in the backoff tests in Section 6.2, the AP uses the traffic it sends to derive the nominal backoff values in order to compare them with the backoffs of user stations. However, if the AP does not have enough traffic to compute these nominal backoffs, it can use the analytical values derived in this appendix.

Our aim is to compute the average actual backoff $E[B_{ac}]$ and the average consecutive backoff $E[B_{co}]$, as a function of the number of contending nodes $n$ (including the AP). We first introduce the *conditional access probability* $\tau$ according to Bianchi's model [10], based on which the *transmission probability* $P_{tr}$ and the *successful transmission probability* $P_s$ are computed. This will help us to compute $E[B_{ac}]$ and $E[B_{co}]$ in sections A.1 and A.2, respectively.

The model described in [10] assumes a saturated channel with all nodes sending CBR traffic.

Based on a two-dimensional Markov chain, Bianchi computes the conditional probability, $\tau$, that a given node accesses the channel at a given time slot:

$$\tau = \frac{2(1-2p)}{(1-2p)(W+1)+pW(1-(2p)^m)} \quad (1)$$

where $p$ is the probability that the transmitted frame collides, $W$ is the minimum contention window size, and $m$ is the number of backoff levels. On the other hand, $p$ is expressed as:

$$p = 1-(1-\tau)^{n-1} \quad (2)$$

For a given $n$, (1) and (2) can be solved numerically (e.g., using Matlab).

Furthermore, the probability $P_{tr}$ that there is at least one transmission in a given time slot can be written as:

$$P_{tr}(n) = 1-(1-\tau)^n \quad (3)$$

This transmission is successful with probability $P_s$:

$$P_s(n) = \frac{n\tau(1-\tau)^{n-1}}{P_{tr}} = \frac{n\tau(1-\tau)^{n-1}}{1-(1-\tau)^n} \quad (4)$$

In the next sections we will use the following relations:

$$\sum_{i=0}^{W-1} x^i = \frac{1-x^W}{1-x}$$

$$\begin{aligned}\sum_{i=0}^{W-1} i\,x^i &= x\frac{\delta}{\delta x}(\sum_{i=0}^{W-1} x^i)\\ &= x\frac{\delta}{\delta x}(\frac{1-x^W}{1-x})\\ &= x\frac{(W-1)x^W - Wx^{W-1}+1}{(1-x)^2}\end{aligned} \quad (5)$$

Assuming that $p(i) = \frac{x^i}{\sum_{i=0}^{W-1} x^i}$ we can write

$$\begin{aligned}\sum_{i=0}^{W-1} i\,p(i) &= \frac{\sum_{i=0}^{W-1} i\,x^i}{\sum_{i=0}^{W-1} x^i}\\ &= \frac{(W-1)x^{W+1} - Wx^W + x}{(1-x^W)(1-x)}\end{aligned} \quad (6)$$

## A.1 Actual backoff $B_{ac}$

The average actual backoff is

$$E[B_{ac}] = \sum_{i=0}^{W-1} i\,p(i)$$

where $p(i)$ is the probability that the actual backoff is equal to $i$ time slots, given that the contention window is $W$ (since all the measured actual backoffs are from the range $[0, W-1]$ as explained in Section 6.2).

$$p(i) = \frac{p_{ac}(i)}{\sum_{i=0}^{W-1} p_{ac}(i)}$$

Now let us compute $p_{ac}(i)$. As shown in Fig. 3, an actual backoff of size $i$ is observed when between two transmissions from the same node $S$:

- we can count $i$ idle time slots (not including DIFS nor SIFS),
- if transmissions occur, they are collision-free.

Hence

$$\begin{aligned}p_{ac}(i) &= [1-P_{tr}(n-1)+P_{tr}(n-1)P_s(n-1)]^i \times\\ &\quad [1-P_{tr}(n-1)]\\ &= [(1-\tau)^{n-1}+(n-1)\tau(1-\tau)^{n-2}]^i(1-\tau)^{n-1}\end{aligned} \quad (7)$$

The first factor in Equation (7) denotes the probability that before any of the $i$ slots, at most one transmission takes place (no transmissions or only one successful transmission). The second factor denotes that none of the $n-1$ nodes (other than $S$) transmits in slot $i+1$.

Let

$$q_{ac} = (1-\tau)^{n-1}+(n-1)\tau(1-\tau)^{n-2}$$

then

$$p_{ac}(i) = q_{ac}^i(1-\tau)^{n-1}$$

and

$$p(i) = \frac{q_{ac}^i}{\sum_{i=0}^{W-1} q_{ac}^i}$$

Therefore, from relation (6),

$$E[B_{ac}] = \frac{(W-1)q_{ac}^{W+1} - Wq_{ac}^W + q_{ac}}{(1-q_{ac}^W)(1-q_{ac})}$$

$E[B_{ac}]$ is compared to the ns-2 simulation results (all nodes are in range of each other; $W = 32$) in Fig. 12. Although both curves have the same shape, there is a difference between them, especially when $n$ increases. The reason is that the analytical model takes into account only collisions due to contention, while simulations consider other causes of collision, such as the shadowing channel model. Hence, in the simulations the probability of observing a collision while measuring a large backoff is higher. This in turn increases the probability of discarding such measurements and decreases the number of large backoff samples, thus resulting in a smaller average. The increasing number of collisions similarly accounts for the decrease of the average value when

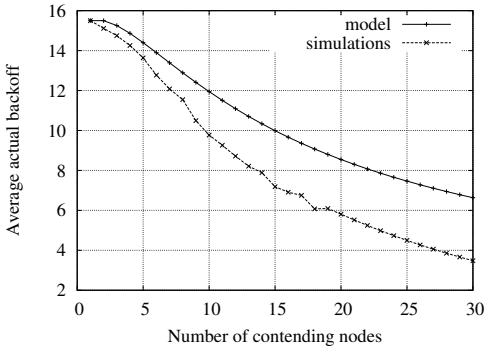



**Figure 12: Average actual backoff.**

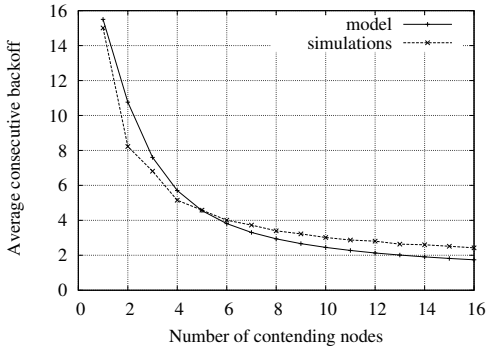



**Figure 13: Average consecutive backoff.**

$n$ increases. In the implementation of DOMINO, the difference between the analytical and simulation values obtained from Fig. 12 can be subtracted from the analytical value in order to get a closer estimate of the real average value.

## A.2 Consecutive backoff $B_{co}$

Using similar reasoning to the one in the previous section, the average consecutive backoff is:

$$E[B_{co}] = \sum_{i=0}^{W-1} i\, p(i)$$

where

$$p(i) = \frac{p_{co}(i)}{\sum_{i=0}^{W-1} p_{co}(i)}$$

$p(i)$ is the probability of the consecutive backoff being equal to $i$ time slots given that the contention window from which this backoff is chosen is $CW_{min}$. $p_{co}(i)$ is the probability of a consecutive backoff of size $i$ unconditioned on the contention window size.

To compute $p_{co}(i)$, consider a successful frame transmission by a node $S$. We obtain a consecutive backoff of size $i$ if and only if none of the $n-1$ other nodes transmits during any of the $i$ time slots nor in the slot in which $S$ transmits. This occurs with probability:

$$p_{co}(i) = [1 - P_{tr}(n-1)]^{(i+1)} = [(1-\tau)^{n-1}]^{(i+1)}$$

Let

$$q_{co} = (1-\tau)^{n-1}$$

then

$$p_{co}(i) = q_{co}^{i}(1-\tau)^{n-1}$$

and

$$p(i) = \frac{q_{co}^{i}}{\sum_{i=0}^{W-1} q_{co}^{i}}$$

Hence, using relation (6),

$$E[B_{co}] = \frac{(W-1)q_{co}^{W+1} - W q_{co}^{W} + q_{co}}{(1-q_{co}^{W})(1-q_{co})}$$

These values are compared to the ns-2 simulation values (with $W = 32$) in Fig. 13. In this case, the two curves are closer than in the previous section. The reason is that consecutive backoffs are smaller than actual backoffs and hence the probability of observing a collision while measuring the former is also smaller than this probability while measuring the latter. Therefore, collisions due to causes not taken into account by the model (such as the channel model) have less effect on the measurement of the average consecutive backoff.