DOMINO: Detecting MAC layer greedy behavior in IEEE 802.11 hotspots

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

EPFL Technical Report IC/2005/025, June 2005

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, [17]).

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). But 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 (Detection Of greedy behavior in the MAC layer of IEEE 802.11 public NetwOrks), a piece of software to be installed in or near the Access Point. DOMINO can detect and identify greedy stations, without requiring any modification of the standard protocol. We illustrate these concepts by simulation results and by the description of a prototype that we have recently implemented.

1. Introduction

IEEE 802.11 [3] wireless LANs were originally meant to be deployed in (relatively) protected locations such as corporate offices; as a result, security, billing, and guarantee of fair access received limited attention. But over the last few years, IEEE 802.11 has also become the dominating solution for hotspots, which provide public wireless access to the Internet.

In this framework, 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 [12], [24] and transport [5] 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 [5] 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 [1], 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 seamless integration in or near the AP[1] without interfering with its normal functions and its ability to identify the cheater.

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 much 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.

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

2. Related work

The problem of MAC layer misbehavior is relatively new and unexplored in the literature. Kyasanur and Vaidya [23] 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

[1]The actual component in which DOMINO has to be installed is the *hotspot controller*, which provides access control and can control several APs [14]; 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.

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.
- 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 [20] 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.

Intrusion detection systems [31] 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., [10], [24]). However, as the problem we consider in this paper focuses on the MAC layer, it is too different to make those solutions eligible here.

In [13] Cagalj et al. study the scenario of multiple cheaters and use game theory to devise optimal cheating strategies. Although their work addresses issues similar to the ones we tackle

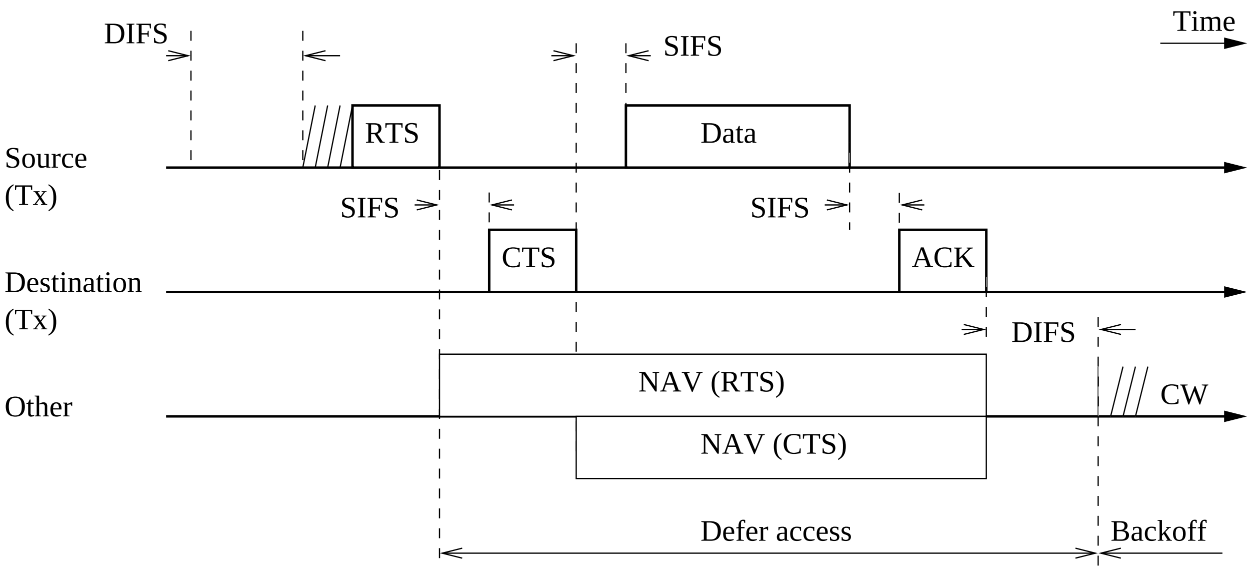


Fig. 1. The distributed coordination function (DCF) of IEEE 802.11 operating in RTS/CTS mode.

here, it does not apply properly in the typical hotspot case.

In [26] we tackled the MAC layer greedy behavior problem and presented the first version of DOMINO; it was limited to attacks on the uplink traffic. In this paper we describe attacks targeted at the downlink traffic and treat additional issues related to detection such as the interaction between DOMINO and the IEEE 802.11e protocol extension for QoS. We also provide related simulation results.

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 an amount of time called DIFS (DCF InterFrame Spacing). It waits for an additional random 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 elapses, 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 of the transmitting station 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.

- DOMINO can be deployed with various APs managed by one or several WISPs. However, for simplicity, 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.
- We consider only the presence of a single cheater. The case of multiple cheaters is addressed - at a theoretical level - in [13].

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

MAC greedy behavior consists in modifying the operation of the IEEE 802.11 protocol by failing to follow communication procedures or changing parameters defined in the standard. Several studies [7], [21] have shown that 91% of the traffic flowing over deployed wireless LANs is TCP and is mainly downlink (i.e., from the AP to the user stations). Hence it is important to distinguish misbehavior techniques according to the type of traffic they target. In the following we describe attacks on the uplink traffic (both TCP and UDP) and the downlink TCP traffic.

4.1.1. Uplink traffic:

- A greedy station can 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, the CW is doubled, and the cheater gets a higher 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 *duration* value in the frame headers in order to prevent the stations in range, that set their NAVs with the value in the *duration* field, from contending during this time. A DoS attack using the same principle was described and evaluated in [9].
    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 8.5.

4.1.2. Downlink traffic:

- In the case of the downlink traffic, the cheater will attempt to increase the share of traffic sent to him through the AP, thus increasing the number of packets destined to it in the AP's queue; to achieve this goal, he will target the protocols responsible for filling this queue. We can distinguish two types of sources (e.g., Web servers) sending traffic to wireless stations through the AP:

  - *UDP source*: since UDP requires no acknowledgements from the receiver and hence cannot be affected by channel conditions, attacking UDP traffic is pointless[2].
  - *TCP source*: on the contrary to the above case, the TCP traffic rate reacts to the channel conditions by using congestion windows and acknowledgements from the receiver. Hence an attack can be mounted on the TCP traffic by exploiting the congestion avoidance mechanism and reducing the source rate until eventually shutting down the flow.

Downlink attacks are relatively less intuitive and require more "effort" from the cheater's side to increase his share of the bandwidth, and from the AP's side to detect the misbehavior. Leveraging on the closed-loop nature of TCP flows, their impact goes beyond the local area (the hotspot and associated nodes) to reach remote servers. Consider the topology in Fig. 2 and the typical following scenario: Two mobile nodes M and Mc are connected to the Internet via the AP. M and Mc download large files from two remote servers, S and Sc, respectively. Both downloads use FTP/TCP. To increase his download data rate, the cheater (Mc) can use the following two techniques to reduce S's data rate, thus freeing more bandwidth for himself at the AP (or at any common bottleneck between the servers and the AP):

- Mc jams the TCP-ACKs from M to the AP, so they never reach the server S. As TCP-ACKs get lost (jammed), S decreases its sending data rate, using TCP congestion control, and ends up killing the connection. At the AP, M's share of the bandwidth decreases, leading to an increase of the data rate from Sc to Mc.
- In the previous technique, the AP can still hear the collisions/jamming and may end up detecting Mc based on the number of retransmissions of M. Another option for Mc consists in jamming AP's frames destined to M, therefore reducing S's data rate, without being heard by the AP. However, Mc's packets share the same queue as M's packets at the AP. While jammed frames get repeatedly retransmitted by the AP, Mc's packets get delayed in the queue, and his data rate (from Sc) decreases as well. To prevent AP's retransmissions and the queueing delays, Mc sends forged MAC-ACKs on behalf of M for the jammed packets[3]. This avoids retransmissions at the AP, while still reducing the data rate from S.

[2]To simplify the discussion, we do not address the case of applications with feedback loops running over UDP

[3]In IEEE 802.11, ACK frames contain no source address fields, therefore making M's task easier.

Furthermore, as we will show in the simulation results, Mc can jam only part of the AP's frames to M, saving his battery power and making detection even harder.

The effects of these misbehavior techniques can be devastating, given the extensive use of TCP in the Internet, especially for Web browsing and file transfers, which are major Internet applications nowadays. It should be noted that using IPsec to encrypt TCP packets will not prevent the cheater from mounting the above attacks because he can simply jam all the packets without distinguishing their content as TCP. He can still distinguish TCP-ACKs by their short size.

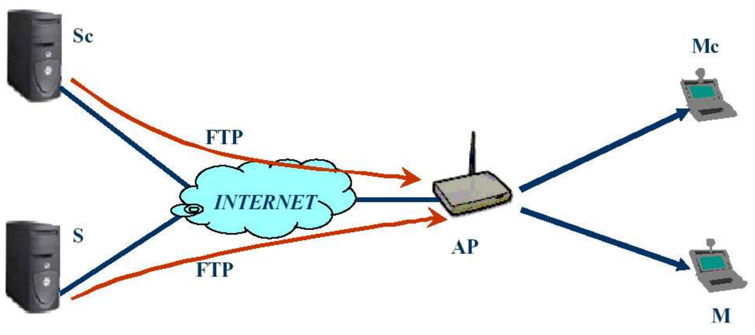


Fig. 2. Generic scenario where Mc jams the AP's TCP packets destined to M (or the corresponding ACKs) in order to reduce the flow from server S.

4.2. Security attacks

This category of attacks (e.g., the deauthentication attack [14]) 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 [14]. As this category has been extensively addressed by other authors, 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. COMPONENTS OF DOMINO

In order to counter the misbehavior techniques presented in Section 4.1, we have devised a detection system dubbed DOMINO. Given the number of possible attacks and their independence,

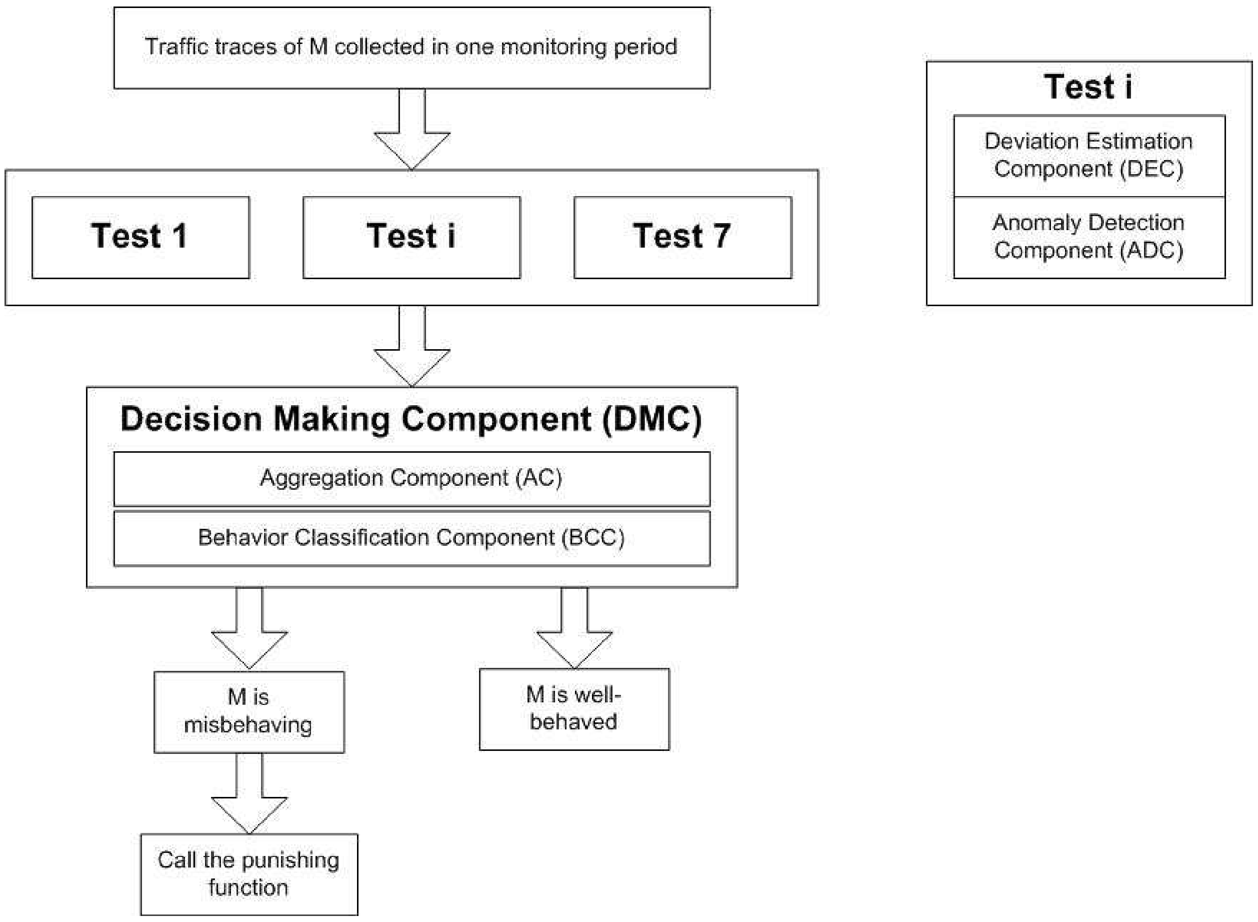

Traffic traces of M collected in one monitoring period
Test 1
Test i
Test 7
Test i
Deviation Estimation Component (DEC)
Anomaly Detection Component (ADC)
Decision Making Component (DMC)
Aggregation Component (AC)
Behavior Classification Component (BCC)
M is misbehaving
M is well-behaved
Call the punishing function

Fig. 3. The modular architecture of DOMINO.

we have adopted a modular architecture, depicted in Fig. 3. As mentioned before, DOMINO needs to be implemented only at the AP.

DOMINO periodically collects traffic traces of active user stations during short intervals of time called *monitoring periods* (the choice of their length is discussed in Section 8.6). A series of *tests*, each aiming at detecting a particular misbehavior technique, determines if the analyzed traffic presents behavior anomalies. The outputs of these tests are then fed into a *Decision Making Component* (DMC) that decides whether a given station is cheating. If so, the control is passed to the misbehavior handling mechanism that, as mentioned before, is dependent on the WISP policy.

The modular architecture presents several advantages. First, the tests as well as the decision making component can be implemented using several algorithms depending on the required accuracy and the tolerable complexity. Second, new tests for potential yet undiscovered misbehaviors can be easily added.

In the following, we present the tests designed to detect the previously presented misbehaviors. Each test consists of two components: a *Deviation Estimation Component* (DEC) and an *Anomaly Detection Component* (ADC). The DEC is typically a statistical test that determines the amount of deviation of a station's behavior, inferred from its traffic trace, from a model of the expected behavior (derived by observing the behavior of the AP or the other active stations during a monitoring period). The ADC uses the deviation measured by the DEC in order to judge a station as well-behaved or suspected. It can be as simple as a comparison of two values or a more complicated technique such as a Bayesian inference.

The DMC aggregates the partial decisions of the different tests in order to assess the behavior of a given station in the last monitoring period. Following the modular approach, we divide the DMC into two modules: an *Aggregation Component* (AC) and a *Behavior Classification Component* (BCC). Again, the implementation of either can be flexible. We have chosen a simple OR of the boolean outputs of different tests to implement the AC. This means that if a station cheats using any of the described methods, it will be detected as cheater. Alternatively, the AC can output a weighted sum of different test outputs; this sum is then normalized to 1 and compared to a threshold. The weights can be chosen to indicate the confidence in a given detection test as well as the severity of the corresponding misbehavior. For example, a test the output of which cannot be affected by factors such as channel conditions would have a higher weight than a test that is more vulnerable to these conditions. Similarly to the ADC, the implementation of the BCC can be based on a simple misbehavior tolerance threshold, or a Bayesian inference.

While the ADCs in different tests can use different or similar implementations (except **Test 7** where we describe a specific ADC), the DEC is specific to each test. Therefore, in the following description of each test, we will focus on the algorithm behind the corresponding DEC. The tests described below use the following structure, where x indicates the test number.

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

```
**if** $condition_x$ is true **then**
    $output_x := 1$
**else**
    $output_x := 0$
```

5.1. "Scrambled frames"

This test aims at detecting misbehavior techniques that rely on frame scrambling; they correspond to the first attacks described in the uplink and downlink parts of Section 4.

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.

Test 1 Scrambled frames

```
$condition_1 : num\_rtx(M_i) < \phi \times E_{j \neq i}[num\_rtx(M_j)]$
```

In this test, $num\_rtx(M)$ is the number of times station M retransmitted its last frame successfully received by the AP. ϕ is a tolerance parameter with a value between 0 and 1; it is applied to the average number of retransmissions of all "other" stations, $E_{j \neq i}$.

DOMINO 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 whether a given frame is destined to itself.

A potential cause of false positives for this test could be the bad channel conditions that lead to frame loss and retransmission. To avoid this pitfall, the AP can take the signal-to-noise ratio of stations into consideration when detecting misbehavior.

5.2. 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 7) 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

```
  $condition_2 : idle\_time\_after\_ACK(M_i) < DIFS$
```

"Oversized NAV"

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

Test 3 Oversized NAV

```
  $condition_3 : A \times actual\_duration(M_i) < duration(M_i)$
```

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 $max_bkf(S_i)$ over a set of frames sent by a given station should be $\geq CW_{min} - 1$, 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 6), 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
  $condition_4 : max\_bkf(M_i) < threshold_{maxbkf}$
```

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.

"Actual backoff"

This test (Test 5) consists in measuring the actual backoff, as shown in Fig. 4. 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, if it is transmitting at low data rates). Then we estimate this backoff by computing the sum as illustrated in Fig. 4.
- 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; in case of collisions, neither the current backoff nor the next one are measured for any station.[4]

```
Test 5 Actual backoff
  $condition_5 : B_{ac}[M_i] < \alpha_{ac} \times B_{acnom}$
```

In Test 5, $B_{ac}[M_i]$ denotes the average *actual backoff* (observed by the AP) of station M_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. If the AP does not have enough data to derive a nominal

[4]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.

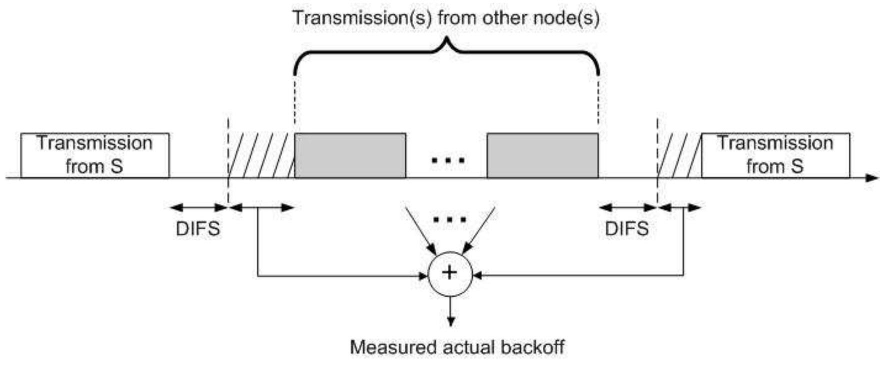


Fig. 4. Measurement of the *actual backoff*. Transmissions from M are interleaved with one or more transmissions from other nodes (including the AP). The transmission includes, in addition to the DATA frame, all the control frames such as RTS, CTS, and ACK, as well as the interleaving idle periods of SIFS and DIFS. The measured value is the sum of all idle intervals (not including interframe spaces) between two transmissions from M.

backoff value from its own traffic, it uses an analytical value $E[B_{ac}]$ (derived in the Appendix). DOMINO does 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 [27] found that, 80% of the time, peak throughput is due to a single user and application, which is typically a large file transfer).

The α_{ac} ($0 < \alpha_{ac} \leq 1$) parameter is configurable according to the desired true positive (correct detection) and false positive (wrong detection) percentages (for example, we use α_{ac} = 90% in our simulations).

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 misbehavior case if 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. 4). The solution to this problem is provided by the *consecutive backoff* test.

"Consecutive backoff"

Fig. 5 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 typically due to

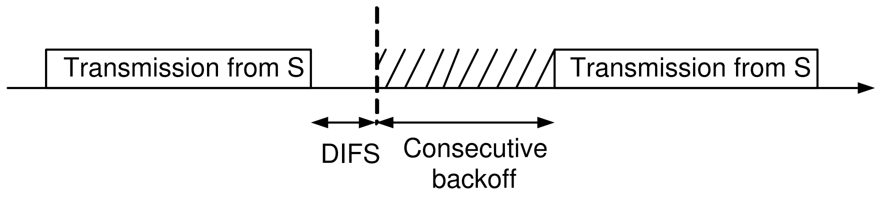


Fig. 5. Measurement of the *consecutive backoff*. Backoff values are taken only between consecutive non-interleaved transmissions from M.

the congestion control of TCP). 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 M sending TCP traffic. We assume that there is enough traffic from other sources on the common channel such that, between two frames sent by M and separated by a transport layer delay, there should be at least one interleaving frame from another station. Hence, if the AP observes two consecutive non-interleaved frames from M, 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 M 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, DOMINO can collect significant samples of the backoff values chosen by M; 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 M can send consecutive non-interleaved frames separated by a delay in addition to the backoff and 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
    $condition_6 : B_{co}[M_i] < \alpha_{co} \times B_{conom}$
```

As with the previous test, the average of the collected values $B_{co}[S_i]$ is compared to a fraction

α_{co} ($0 < \alpha_{co} \leq 1$, we use α_{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).

Scrambled TCP packets with forged MAC ACKs

The second downlink attack (Section 4.1) is the most sophisticated. It is also the most difficult to detect because the AP cannot hear collisions and, since the cheater forges the MAC ACKs corresponding to scrambled frames, DOMINO cannot rely on the number of retransmissions to detect misbehavior. To cope with this technique, we have devised two complementary mechanisms that implement the DEC and ADC components of the test in the system architecture. First, DOMINO measures the throughputs of the downlink flows (this is the DEC). Then, if there is a receiver that draws most of the traffic, DOMINO suspects it as a potential cheater. As will be explained in Section 8.1, throughput is not a reliable detection metric because of the different needs of users. Hence, we use *Dummy Frame Probing* (DFP) to confirm the suspicion or reject it. DFP consists in sending dummy frames to virtual nonexisting stations. If any of these frames is followed by a MAC ACK, this is an indication of an existing cheater in the network. Longer throughput observation is then needed to determine the identity of the cheater. DFP combined with throughput comparison constitute the ADC.

A clever cheater can construct the list of virtual stations (by recording stations that do not reply with a MAC ACK) in order not to respond to the dummy frames. To detect this cheater, the AP should also generate fake ACKs: in this way, the cheater cannot easily distinguish the dummy frames from the other ones. But for the cheaters it is beneficial to attack only the connections with high throughput. Thus, in order to be effective, the dummy frames must be generated from time to time at high throughput as well, as a trap for the cheater. The advantage of dummy frames is that they represent a highly discriminating test: a simple sample is enough to raise a very high suspicion, even if they are generated during a small amount of the time (e.g., 5 to 10%). Thus, the resulting overhead is small.

6. SIMULATION RESULTS

In order to study the performance of the proposed solution, we have used ns-2 [15] to simulate DOMINO. 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

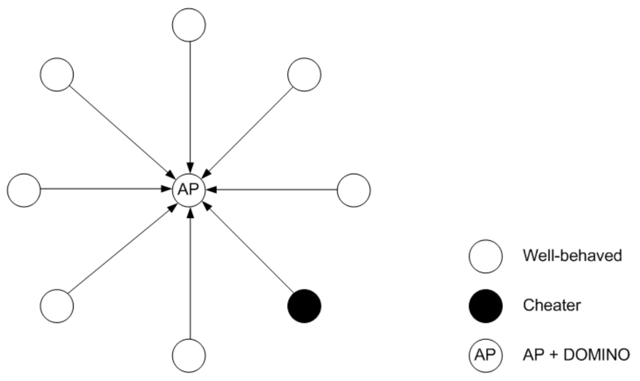


Fig. 6. Simulation scenarios: 8 stations send UDP or TCP data to the AP, which also generates traffic similarly to one station and sends it to an additional receiver station (not shown in the figure). The distance between each station and the AP is 50m. All stations are within radio range of each other.

complete detection mechanism.

6.1. Uplink traffic

6.1.1. 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. 6). Due to lack of space, we cover only these scenarios that represent common traffic types.

1) UDP traffic

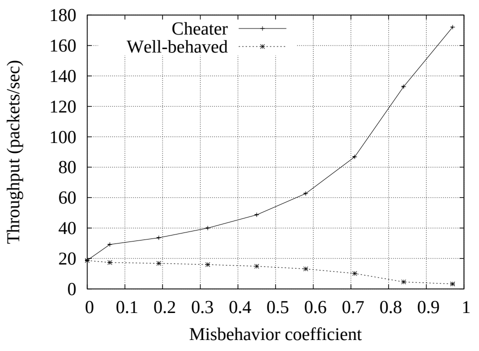
   Besides the cheater, there are 7 stations sending CBR traffic (the nominal rate is 500 bytes/packet, 200 packets/s); the cheater is also a CBR source. The cheating technique consists in decreasing the contention window.

   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.

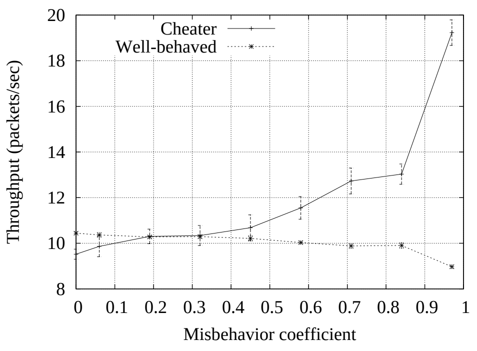
2) TCP traffic

   Each of the 8 stations runs an FTP application; one station is cheating by jamming TCP packets and forging the corresponding MAC ACKs.

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



(a) UDP traffic



(b) TCP traffic

Fig. 7. Throughput comparison between misbehaving and well-behaved stations.

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, represented by the following equation:

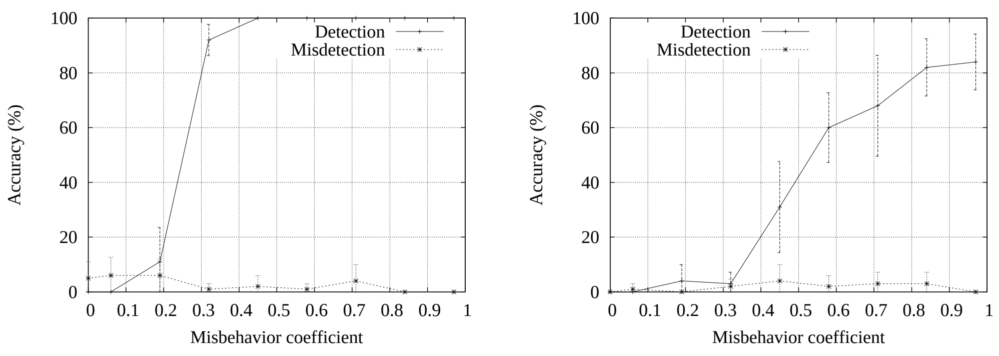
$$\left[\frac{P_r(d)}{P_r(d_0)}\right]_{dB} = -10\beta log\left(\frac{d}{d_0}\right) + X_{dB}$$

where $P_r(d)$ is the mean received power at distance d, d_0 is a reference distance, β is the path loss exponent, and X_{dB} is a Gaussian random variable with zero mean and standard deviation σ_{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, where measurements are not taken into account in the results.

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.

6.1.2. Impact of misbehavior on throughput: Before presenting the performance of DOMINO, we compare the throughput values of cheating and well-behaved stations in both simulation scenarios.



(a) *Actual backoff* test in the UDP traffic case. (b) *Consecutive backoff* test in the TCP traffic case.

Fig. 8. Performance of the *Actual backoff* and *Consecutive backoff* tests.

Fig. 7 shows that MAC misbehavior results in throughput[5] 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.

6.1.3. 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. 8(a), 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 DIFS).
- 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.

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

6.1.4. 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 5.2) 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(b) 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.

6.1.5. Complete mechanism: The descriptions of the *actual backoff test* and the *consecutive backoff test* in Section 5.2, 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.

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.

6.2. Downlink traffic

In this section we only simulate the second of the two attacks on the downlink described in Section 4.1, i.e., referring to Fig. 2, Mc jams the TCP packets from server S to M, transmitted by the AP, and sends MAC-ACKs on behalf of M.

In order to save energy and to make detection harder for the AP, Mc may jam only a proportion X of the downlink traffic (e.g., when X=1, Mc jams all frames from the AP to M). This proportion of jammed packets can be either uniformly distributed in time, or applied in bursts. In the latter case, Mc jams the channel during D seconds, for each period T with $D < T$ (therefore $X = D/T$). We refer to this method as “bursty jamming”.

The performance metrics we analyze are the throughputs of the cheater (Mc) and the well-behaved node (M).

Sources S and Sc start transmitting at the same time, using TCP-Reno with 1000-byte packets. To make sure that the throughput reached a steady state, we let the cheater start jamming 60 seconds later. Results are averaged over 35 simulations. Three factors help Mc increasing his throughput:

- Reducing the number of competing flows entering the queue at the AP
- Reducing the collision rate on the wireless channel (TCP-DATA transmitted by the AP with TCP-ACKs transmitted by M and Mc)
- Reducing the queueing delays at the AP (the jammed packets are not retransmitted)

Figure 9(a) shows the throughput of Mc and the throughput received by M when we vary the proportion of jammed frames X. We can see that jamming 30% of the frames is enough to reduce M’s received throughput to zero, and increase Mc’s received throughput to the maximum available data rate. The evolution of Mc’s throughtput and M’s received throughput in time is shown in Fig. 9(b). Note the low throughput of Mc when he first starts jamming M’s frames and forging MAC-ACKs. Later on, this overhead is reduced since M receives decreased throughput, therefore Mc jams less frames and forges less MAC-ACKs, increasing its efficiency. This transient period lasts less than 10 seconds.

Figure 10 shows the same metrics when the cheater applies bursty jamming. Using T=1s, inspired from [22], [4], the same jamming $D/T = X$ proportion leads to a better throughput for Mc and lower consumption for M making the hybrid attack even more devastating.

7. Implementation

To prove the need for and the efficiency of the proposed detection system, we have implemented one of the cheating techniques, based on backoff manipulation, and a prototype of DOMINO. We used a simple scenario where two stations are uploading UDP traffic to the AP.

We have performed experiments corresponding to several values of the cheater contention window, which should be of the form $2^n - 1$, where n is an appropriate integer. Specifically, we

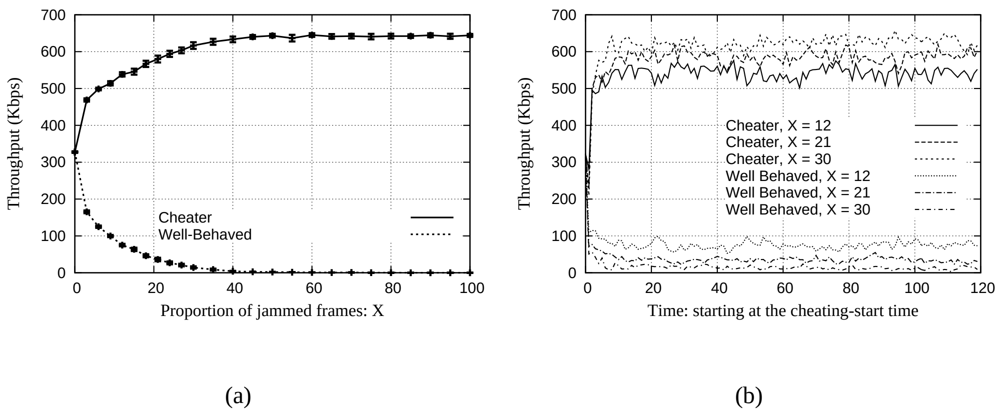


Fig. 9. Cheating performance using regular jamming on downlink traffic.

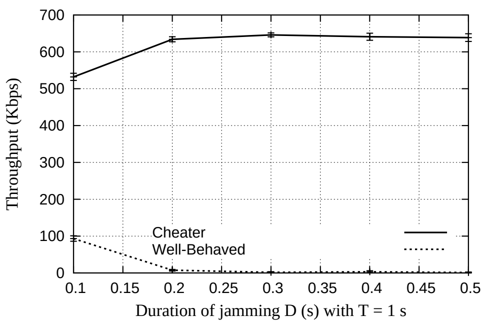
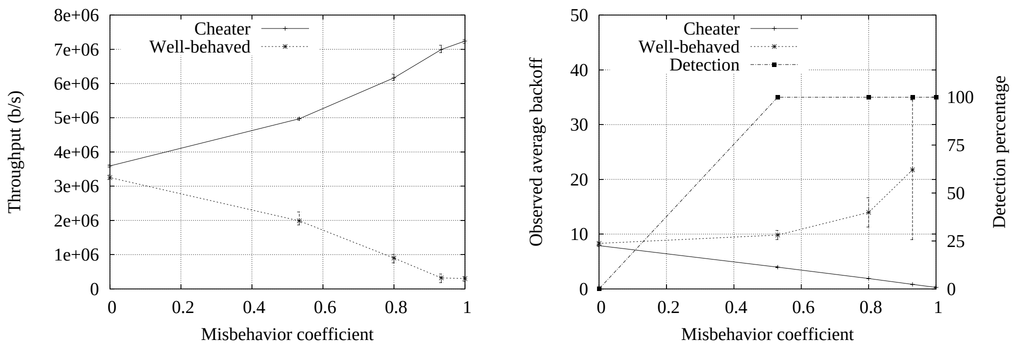


Fig. 10. Cheating performance using bursty jamming on downlink traffic.

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



(a) Throughput (b) Backoff

Fig. 11. Experimental results. On the x-axis, the *misbehavior coefficient* takes values: 0, 0.53, 0.8, 0.93, and 1.

the misbehavior percentage (due to collisions and the subsequent increase of the contention window); this explains its decreasing throughput.

8. Discussion

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

8.1. Throughput as a detection metric

Although throughput seems to be the most intuitive metric for distinguishing stations using higher shares of the channel bandwidth than other stations, it cannot be used as a metric.

Indeed, 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 to the AP).

Experimental studies in [6] and [30] have shown that the throughput of a UDP source in a wireless network is affected by many factors, such as packet overhead, SNR, network and host hardware, device drivers, and network protocol implementations in the operating system. The authors of [16] 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 close to that of the disadvantaged station. The negative effect of SNR on channel capture is explored in [28] (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 [30]. 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 [8], [19], [25] typically using Jain's fairness index [18] (which in turn uses channel bandwidth shares), throughput is far from being the optimal misbehavior metric, in our case.

8.2. Hidden terminals

Hidden terminals may have a negative effect on DOMINO. 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. We have rerun both simulation scenarios for uplink traffic (UDP and TCP) with hidden terminals (by changing the reception range) and by choosing appropriate values for detection thresholds (specifically, α_{ac} and α_{co} defined in Section 5.2), i.e., by tolerating some misbehavior, we have managed to reduce false positives in the presence of hidden terminals.

8.3. IEEE 802.11e for quality of service support (whole subsection added)

The increasing user requirement for multimedia support and QoS in wireless networks has motivated many research and standardization groups to work on satisfying these needs. In particular, IEEE 802.11 Task Group E is developing a new 802.11 standard with QoS support, called IEEE 802.11e [29], expected to be released soon. The detailed description of 802.11e is out of scope of this paper, thus we restrict our short description to the aspect of relevance to our cheating detection mechanism.

Enhanced DCF (EDCF) defines 8 different priorities mapped to 4 access categories ($AC[i]$). These access categories correspond to 4 different queues (Fig. 12) assigned with different MAC parameters: initial contention window values ($CW_{min}[i]$), maximum contention window values ($CW_{max}[i]$), and arbitration inter-frame spacings ($AIFS[i]$). The AIFS of IEEE 802.11e plays a role similar to DIFS in the legacy IEEE 802.11 standard.

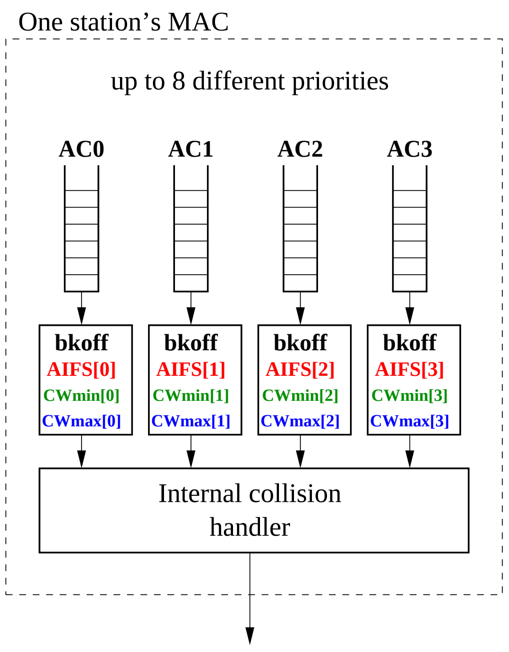


Fig. 12. IEEE 802.11e MAC for QoS support.

Before being transmitted on the wireless channel, a packet from $AC[i]$ contends with other packets, from other queues ($AC[j], j \neq i$) in the same station. This internal contention is similar to the one between different stations in the legacy 802.11. The packet that *wins* the internal contention gets to contend with other packets from other stations to access the wireless channel. Among these internal and external contentions, a monitor (e.g., DOMINO) only sees the latter. Taking into account the above description, DOMINO faces two challenges on which we are currently working:

- It is not clear which value to monitor. For example, there are different backoffs of different priorities, of which the monitor is unaware.
- The values to which comparison should be made are not well defined since the AP does not necessarily have the same combination of priority flows as other nodes, it will not be able to compare their measured access parameters (backoffs) to its own parameters. Deriving an

analytical model for all possible traffic combinations is another challenging problem.

8.4. 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 [14], 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 [14], 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. In addition, 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 enhanced security mechanisms jointly with DOMINO. We will consider the details of this solution in our future work.

8.5. 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 DOMINO 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.

8.6. 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 in [8], [19], [25] 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 rely on 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. 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. Once a hacker has implemented an attack, he can make it available on a web site, thus jeopardizing the proper operation of many wireless networks around the globe.

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.

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 a shared medium.

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.

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 Petros Zerfos for their helpful feedback on earlier versions of this work.

REFERENCES

[1] `http://domino.epfl.ch`.

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

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

[4] Imad Aad, Jean-Pierre Hubaux, and Edward W. Knightly. Denial of service resilience in ad hoc networks. In *Proceedings of ACM Mobicom*, Philadelphia, Pennsylvania, USA, 2004.

[5] 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.

[6] 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.

[7] M. Balazinska and P. Castro. Characterizing mobility and network usage in a corporate wireless local-area network. In *Proceedings of the International Conference on Mobile Systems, Applications, and Services (MobiSys)*, May 2003.

[8] 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.

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

[10] 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.

[11] 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.

[12] 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.

[13] M. Cagalj, Saurabh Ganeriwal, I. Aad, and J. P. Hubaux. On selfish behavior in csma/ca networks. In *Proceedings of IEEE Infocom*, Miami, Florida, USA, March 2005.

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

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

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

[17] 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.

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

[19] 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.

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

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

[22] A. Kuzmanovic and E. Knightly. Low-rate TCP-targeted denial of service attacks (the shrew vs. the mice and elephants). In *Proceedings of ACM SIGCOMM 2003*, Karlsruhe, Germany, August 2003.

[23] Pradeep Kyasanur and Nitin Vaidya. Selfish mac layer misbehavior in wireless networks. *accepted for publication in IEEE Transactions on Mobile Computing*, April 2004.

[24] 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.

[25] 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.

[26] M. Raya, J.P. Hubaux, and I. Aad. DOMINO: A system to detect greedy behavior in IEEE 802.11 hotspots. In *Proceedings of MobiSys'04*, June 2004.

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

[28] 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.

[29] IEEE 802.11 WG. *IEEE 802.11e/D4.3, Draft supplement to part 11: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Medium Access Control (MAC) Enhancements for Quality of Service (QoS) IEEE Std. 802.11e/D4.3*. IEEE, 2003.

[30] 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.

[31] 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 5.2, the AP observes 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* τ according to Bianchi's model [11], 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 [11] assumes a saturated channel with all nodes sending CBR traffic.

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

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

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:

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

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:

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

This transmission is successful with probability P_s:

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

In the next sections we will use the following relations:

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

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

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

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

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 5.2).

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

Now let us compute $p_{ac}(i)$. As shown in Fig. 4, 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

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

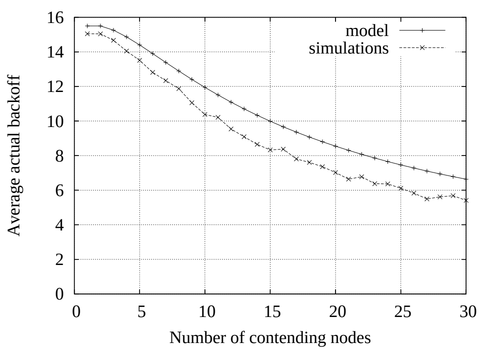
The first factor in Equation (7) denotes the probability that before any of the i slots, at most one transmission takes place (no transmissions or only one successful transmission). The second factor denotes that none of the $n-1$ nodes (other than S) transmits in slot $i+1$.

Let

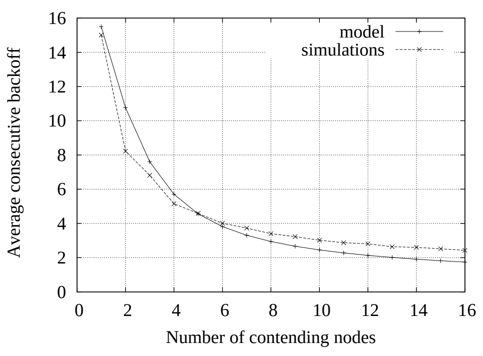
$$q_{ac} = (1-\tau)^{n-1} + (n-1)\tau(1-\tau)^{n-2}$$

then

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



(a) Average actual backoff



(b) Average consecutive backoff

Fig. 13. Analytical and simulation models of average backoff.

and

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

Therefore, from relation (6),

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

$E[B_{ac}]$ is compared to the ns-2 simulation results (all nodes are in range of each other; $W = 32$) in Fig. 13(a). The increasing number of collisions accounts for the decrease of the average value when n increases. In fact, the more stations contend to access the channel, the shorter the observed backoff will be (accounting for successful transmissions). Large backoff samples (following collisions) are discarded from the observations since the AP cannot identify the stations involved in collisions, as explained earlier. In the implementation of DOMINO, the difference between the analytical and simulation values obtained from Fig. 13(a) 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),

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

These values are compared to the ns-2 simulation values (with $W = 32$) in Fig. 13(b). Note that, for the same reasons as for the actual backoff measurements, the average consecutive backoff decreases when the number of contending nodes increases. Furthermore, the occurrence of consecutive backoffs is much less frequent than the occurrence of actual backoffs, therefore it needs longer observation periods.