A Wired-side Approach to MAC Misbehavior Detection

Aravind Venkatarama
Department of Computer Science
Georgia State University
Atlanta, GA 30303
aravindv85@gmail.com

Cherita Corbett
Applied Physics Lab
Johns Hopkins University
Laurel, MD 20723
cherita.corbett@jhuapl.edu

Raheem Beyah
Department of Computer Science
Georgia State University
Atlanta, GA 30303
rbeyah@cs.gsu.edu

***Abstract* - We propose a simple scheme for detecting selfish behavior achieved by manipulating the 802.11 Medium Access Control (MAC) protocol. Specifically, attacks that exploit the Distributed Coordination Function (DCF) parameters and data rate adaption scheme to maximize individual throughput pose a denial of service threat against protocol abiding nodes. We detect this malicious behavior by employing a combination of supervised and unsupervised learning techniques that monitor for disparities in the delay patterns of protocol-abiding and illegitimate traffic. Unlike existing approaches, detection is done on the wired side. We apply an anomaly-based categorization, which obviates the need to train on traces from different network instances. Since the approach is holistic and does not rely on a feature selection using individual parameters, the technique is free of adaptive cheating. Additionally, the accuracy of classification is independent of the number of terminals in the network, the number of colluding attackers, protocol, rate adaptation and higher layer transmission behavior. Simulations and experiments are used to validate our scheme.**

***Index Terms* – MAC misbehavior, 802.11 MAC protocol, Distributed Coordination Function.**

I. INTRODUCTION

As a considerable portion of 802.11 wireless driver functionality shifts to software with the goal of increased customization, it becomes easier to cheat at the MAC layer by exploiting medium access vulnerabilities. As a result, the inherent fairness in the MAC protocol is removed to maximize channel utilization for the malicious node. Noted ways of performing this include tweaking DCF parameters (contention window, slot time, SIFS, NAV, CTS/RTS thresholds), scrambling frames and intentionally colliding with external CTS/RTS frames.

In this paper, we do not consider attacks that target specific frames. We present an abstract methodology for detecting DCF parameter manipulation as a whole. Further, we illustrate a new technique for cheating by disabling rate adaptation and detect this form of misbehavior as well.

Motivation for cheating at any layer is bandwidth gain when sharing a medium with others. Hence, we address the problem in an infrastructure wireless setting as opposed to an ad hoc network, strategically performing the detection on the wired side. The core of our detection scheme is an agent *D* (Figure 1) sitting atop a switch, or a separate monitoring

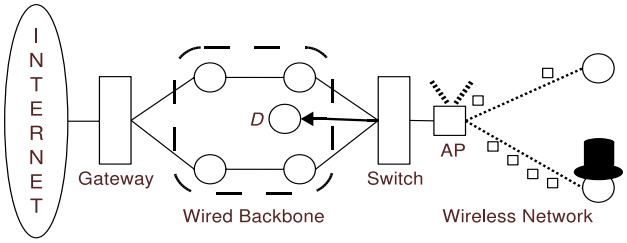

Fig. 1. MAC Misbehavior: Illustration.

device that is connected to the mirror port of a switch, that passively sniffs passing traffic streams on the wired side and observes the influence of misbehavior on packet inter-arrival times (IATs). To this end, we propose a novel architecture that can be implemented on the wired side of the network, that detects attacks by sampling incoming traffic streams and correlating packet IAT distributions.

Our misbehavior classification technique is primarily based on an anomaly-based classifier that monitors for exceptions from the normal delay of nodes that do not cheat on the DCF. The classifier makes no assumption on the distribution of the normal delay values and is not trained on a predefined behavior. Instead it works by seeking a deviation in the *closeness* of incoming IAT sequences. It runs a Bayesian test at runtime to do so.

The proposed technique is a simple solution that does not require procedurally intensive functionality to be implemented on wireless terminals/access points or modifications to the 802.11 standard. As all wireless traffic through the base station passes (or can be routed) into the wired backbone where detection takes place, the scheme is centralized and is not affected by issues that accompany wireless-side detection, such as, interference, collisions, visibility and scalability. Since the primary classifier performs a relative, basis-less analysis of incoming traffic to identify misbehaving traffic, attackers cannot make subtle adaptations to their routine and sneak under the radar. Also, it is not limited to 'available' signatures and can detect patterns that may be missed by analytical feature set generators. This is important, because depending on the number of nodes, collision probability, protocol, rate adaptation and other similar influential factors, there may be a lot that a purely supervised classifier does not account for. We also take into account the scenario of colluding attackers, where a group of

malicious individuals or single-user controlled bots could target a well-behaved network in an attempt to cause a network-wide denial of service.

The remainder of the paper is organized as follows. Section II outlines previous work broadly classifying them into three categories based on techniques used. Section III provides an analysis on the effect of cheating at the MAC layer. The proposed detection technique is discussed in Section IV. We discuss the scheme's scalability in Section V. Accuracy evaluations are given in Section VI. Section VII gives the conclusion and future work.

II. Related Work

The literature covered in this section is primarily concerned with misbehavior techniques that exploit MAC fairness. Works that discuss denial of service attack models in WLANs that exploit other 802.11 vulnerabilities are not directly relevant in the present context and are not addressed. Current work on MAC misbehavior can be broadly classified into three categories. The first category consists of approaches that analytically reproduce "random" back-off in an attempt to emulate the idle time between legitimate transmissions. They try to extract a deterministic behavior model from a stochastic system in order to recreate expected base-line profiles. The second category assumes that fabricating back-off values is not scalable and proposes changes that incorporate detection in wireless nodes. The third category focuses on the effect of misbehaving senders in the absence of an arbiter in ad hoc networks.

References [1-2] fall in the first category. In [1], the proposed method requires wireless nodes to monitor the idle time between a successful transmission and the subsequent RTS from and to their immediate neighbors. Based on collision probability (p_c), nodes analytically construct profiles for legitimate terminals' distributions to be compared against unknown traffic. Nodes calculate p_c from the frequency of collisions as observed in their vicinity. The method in [2] assumes that the network is saturated. It uses a Markov chain based model to determine the IAT distribution, complete with consideration for p_c. However [1-2] do not account for the fact that in addition to depending on the system state (e.g., number of terminals at a given instance), p_c is also a function of frequency and duration of transmissions, which depend on higher layers. Furthermore, [2] assumes that nodes transmit at a constant rate and does not take into account the influence of rate adaptation on legitimate traffic.

In the second category, [3-5] suggest changes to be made to either the driver or protocol to include detection into the 802.11 wireless architecture. The authors of [3] propose a Predictive Random Back-off algorithm that tunes the Binary Exponential Back-off algorithm to generate a reproducible back-off that can be monitored for misbehavior. In [4], the receiver assigns back-off values to the sender. The receiver assigns an initial back-off from which the sender calculates a new back-off as a function of the assigned back-off and number of retransmissions. On receiving data from the sender, the receiver calculates the new back-off based on the number of retransmissions to check for misbehavior. This active method requires changes to be made to the driver and adds computational overhead as well as redundancy on the receiver and sender sides in calculating the new back-off. In [5], detection is done at the access point and involves a series of tests to check for misbehavior on different levels. However, it uses the magnitude of statistics, such as mean of back-off, as the primary metric for classification. This is not advisable as the attacker may adjust the back-off sequence in a way that the effective mean equals the expected. Also, it monitors the number of idle slots, which means that if the attacker cheats on slot time and not on contention window, he would not be detected because the mean number of idle slots stays constant.

The third category [6-10] includes studies that analyze the consequence of misbehavior in ad hoc networks. References [6-8] work on similar lines where individual nodes monitor their neighbors' back-off. The model in [6] assumes that the monitoring node follows the same back-off sequence as its neighbors and anticipates the same behavior from them. In [7], the sender and receiver strike a mutual agreement on the expected back-off values at the beginning of the transmission. This model works on the logic that as long as one of the two nodes is honest, the system is free of cheaters. In [8], *tagged* nodes announce the state of their pseudo-random generator (PRNG) upon which monitoring neighbors determine the expected sequence. Since the MAC address is used as a seed for the PRNG, MAC spoofing can result in others getting caught while the attacker remains undetected. The above solutions do not account for interference and hidden terminal problems.

Our technique for misbehavior classification is free of the above mentioned problems as it performs a basis-less comparison of unidentified traffic distributions as opposed to a comparison with analytically constructed *legitimate* profiles. Misbehavior detection is heuristic and anomaly-based. Our technique is independent of rate adaptation and higher layer behavior unlike the first category, does not require changes to the protocol/driver unlike the second category, and does not seek to address misbehavior in ad hoc networks as the related work in the third category does. Also, to the best of our knowledge, we are the first to propose wired-side detection of 802.11 MAC misbehavior. Since only the successful transmissions from the WLAN carry over to the LAN, wired-side detection offers a clean, definitive solution that is not hampered by noise and hidden terminal problems. As it offers a single point of discovery, it is also scalable. The proposed idea is rather simple, too. It relies on the fact that for a malicious node to be successful, its IAT distribution will look different from existing traffic. Thus we are able to distinguish between malicious and legitimate traffic by using adaptive learning techniques.

III. ANALYSIS

A. Misbehavior study

This section illustrates the three kinds of misbehavior addressed in this paper, highlighting the consequence on legitimate terminals. *Cheating on DIFS*, that is, transmitting a frame before the default DIFS interval expires, reduces the minimum delay incurred by a wireless node. *Cheating on the contention window (CW)* shrinks the random back-off period that follows DIFS. Cheating on DIFS and/or the contention window allows an attacker to have a shorter inter-arrival time (Figure 2b) than a legitimate node that conforms to the correct operation of the protocol (Figure 2a). *Cheating on rate adaptation* involves transmitting at the maximum available data transmission rate (e.g., 11Mbps for 802.11*b*) by turning off Auto Rate Fall-back (ARF) [12], while the legitimate nodes adapt to lower data rates to handle packet loss. Given the sub-optimal operation of ARF [13], it may be advantageous to switch off rate adaptation to transmit faster.

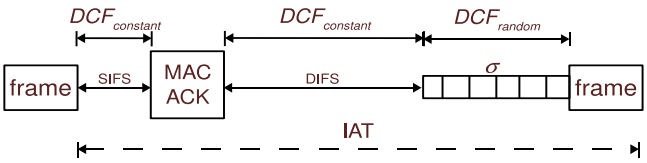

Fig.2. (a) DCF working: Legitimate.

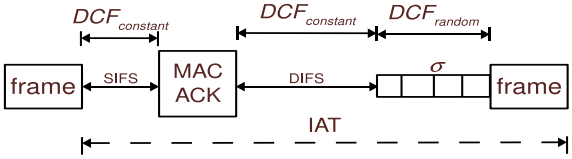

Fig. 2. (b) DCF working: Attacker.

To demonstrate the effects of cheating with DIFS and the contention window, we used the simulation setup described in Section V and measured the distribution of the DCF portion (that is, $DCF_{constant}$ + DCF_{random} from Figure 2) of the IAT associated with packets within a trial.

In the case of DIFS cheating, we considered an extreme attack scenario (Trial I), where we set the DIFS parameter of the attacker to 14*us*; and for a moderate scenario (Trial II), we set the DIFS parameter to 30*us* for the attacker. The legitimate node's DIFS was left at the prescribed 50*us* during both trials. Figure 3a clearly shows the disparity in the delay of the DCF between the attacker and the legitimate node for both trials.

Similarly, we measured the effects of cheating with the contention window for an extreme ($CW_{min} = CW_{max} = 2us$) and moderate ($CW_{min} = CW_{max} = 16us$) scenario. For both scenarios the legitimate node's contention window was left at the default setting ($CW_{min} = 32us$, $CW_{max} = 1024us$). Figure 3b shows the effect of cheating on the contention window.

While the first two kinds of cheating have been studied before, we introduce the third. As part of the IEEE 802.11 standard, wireless nodes perform rate adaptation (shifting to a

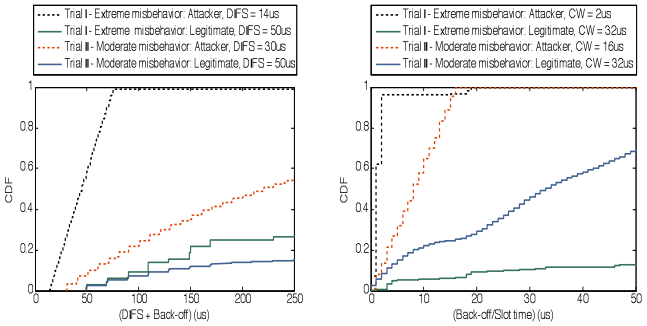

Fig. 3. (a) DIFS cheating, (b) CW cheating.

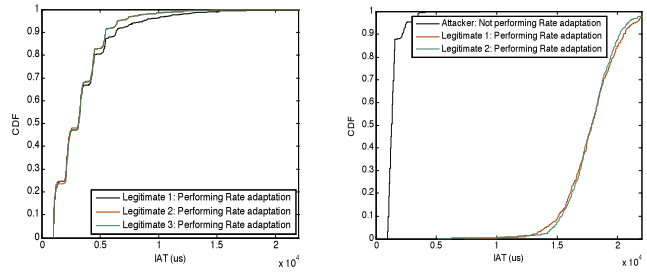

Fig. 4. (a) Rate adaptation cheating: No attack., (b) Rate adaptation cheating: Under attack.

lower/higher optimal PHY data rate) to improve performance during physical layer interference (e.g., radio frequency interference). However, the most widely deployed rate adaptation algorithm, ARF, is often inappropriately invoked as a result of packet collisions at the MAC level. This inappropriate functionality, interestingly enough, actually creates a level of fairness in the network as the nodes share the inappropriate penalty by switching to lower transmission rates as a result of MAC layer collisions. This phenomenon is illustrated in Figure 4a. The overlap of the IAT distributions illustrates the fairness that results because of DCF as well as all nodes enabling rate adaptation. In our misbehavior scheme, an attacker stands to benefit with increased bandwidth utilization by simply not performing rate adaptation (Figure 4b) and continuing to transmit at the fastest physical layer data rate. Figure 4b shows that the distribution of the attacker's IAT is less than the legitimate nodes (indicating a higher sustained throughput), illustrating that nodes not employing rate adaptation (though flawed in its most widely deployed form - ARF) can take away bandwidth from legitimate nodes. Note that this works to the attacker's interest when there is little physical layer interference as not adapting the rate in a scenario with increased physical layer interference may result in the attacker's packets being too mangled to read at the receiver thus reducing his throughput. It should be mentioned that though ARF is the most-widely deployed rate adaptation protocol, there have been many proposed schemes [14-16] to deal with this protocol's inefficiency.

B. Preliminary experiments

As detection on the wired side is a distinguishing factor of

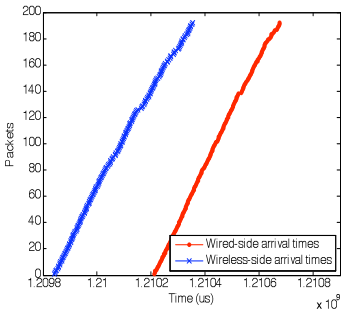

Fig. 5. Packet arrival times on wired and wireless sides.

this approach, in this section we illustrate its viability. We conducted experiments to determine whether the temporal characteristics of the IAT observed on the wireless link were in tact on the wired side. This is a function of the queuing delay and the number of router hops in the path to the destination.

For this purpose, an experimental testbed was built using three Lenovo laptops, three Dell desktops, a Netgear 10/100 Mbps Fast Ethernet switch and a Linksys 2.4Ghz wireless-*b/g* access point (AP). The setup is similar to the attack scenario depiction in Figure 1. The classifier monitors at the switch immediately linking the access point to the LAN.

The arrival times of upstream traffic were recorded at the receiver node on the wired side. On the wireless side, a laptop acting as sniffer was used in promiscuous mode to capture traffic from the wireless sender. We observed that the arrival rates were retained albeit with a uniformly witnessed lag (as a result of queuing in switch) as shown in Figure 5 above. Note that given the simple one-hop path from the WLAN to the classifier node on the wired side, a switch with minimal traffic density exhibits a constant queuing delay.

IV. Detection Scheme

A. Misbehavior detection

While some related work perform a signature-based categorization, our classifier compares the delay distributions of unknown sequences with each other to find the *closeness* '*c*'. Each subsequent trace is examined for likeness to the preceding incoming traces. A hypothesis test is performed over a threshold of likelihood (c_{thresh}) that the classifier is initially trained on, but is dynamically updated as the classifier learns. The motivation behind using an anomaly-based detection scheme is that during a given time window, all legitimate nodes in a network are expected to converge to a similar timing profile. This will be shown in Section V.

B. Classification Scheme

We bin IAT values into *n* bins (*n* depends on the bin width) and calculate for each dataset the number of occurrences in each bin to generate a profile *f*. To detect misbehavior, we use a Naïve Bayes classifier to compare profiles of the first trace with those of every other trace to calculate *c*. This is done using a two-sample Chi-square test.

Profiles f_i are compared with an unknown sample f_x based on frequency of occurrences in each bin. Because the nature of incoming traffic cannot be predicted, prior probability is unknown and is assumed equally distributed over the *n* profiles.

$$PriorProbability \quad P(f_i) = 1/n \tag{1}$$

$$Likelihood \quad P\langle f_x | f_i \rangle \tag{2}$$

$$PosteriorProbability \quad P\langle f_i | f_x \rangle = P\langle f_x | f_i \rangle . P(f_i) \tag{3}$$

Since f_x is a random variable $\{x_1, x_2, \ldots x_d\}$,

$$P\langle f_i | f_x \rangle = P\langle (x_1, x_2, \ldots x_d) | f_i \rangle . P(f_i) \tag{4}$$

$$P\langle f_i | f_x \rangle = P(f_i) . \prod_{k=1}^{d} P\langle x_k | f_i \rangle \tag{5}$$

Likelihood (measure of how similar the unknown trace is to a given profile) is calculated for each profile using a two-sample Chi-square test, which is run independently on all sample-profile bin frequency pairs. Posterior probability (measure of how likely a profile is the closest match for the unknown) is derived by aggregating the Likelihood measures (Chi-square values) each of which is calculated as shown below.

$$\chi^2 = \sum_{i=1}^{k} \frac{\left(S_{1_i} - S_{2_i}\right)^2}{S_1 + S_2} \tag{6}$$

S_1 and S_2 are bin frequencies of the two samples to be compared. They represent an unknown and a sample profile. *k* is the number of bins.

V. Performance Analysis

A generic simulation template of up to 50 wireless nodes was created in *ns2* [17], each of which could choose from different traffic models, transport protocols, WLAN speeds, locations, mobility patterns and packet sizes. MAC parameters were altered at the selfish nodes, while default parameter values of the 802.11*b* standard were retained at legitimate nodes. Table 1 shows the MAC parameters that were used during the simulations, from moderate to extreme cheating. The default values for (CW_{min},CW_{max}) and DIFS in the 802.11*b* amendment are (32,1024) and 50μs respectively.

Though simulations were performed for a longer duration, individual 10-second time windows were monitored for anomalies. A 1000*1000 m^2 sized topology was created as the testbed backbone. The setting includes receiver nodes on the wired side and a base station. TCP and UDP traffic generation involved the use of FTP and CBR applications. Node location changes were performed using varying levels of two-dimensional spatial placements within fixed coordinates. The Random Waypoint mobility model was used for node movement.

In the simulation setup, several parameters were varied to study their effect and to check the robustness of our concept. The parameters that were varied include the number of nodes, data rate, transport protocol, location, mobility and traffic model. The results in this section are a representative sample of

Table 1: MAC Parameters for selfish nodes

MAC Parameters
Class A: $(CW_{min}, CW_{max}) \in \{(1,1), (2,2)\}$ *OR DIFS* $= \{1\text{-}10\}$
Class B: $(CW_{min}, CW_{max}) \in \{(4,4), (8,8)\}$ *OR DIFS* $= \{11\text{-}25\}$
Class C: $(CW_{min}, CW_{max}) \in (16,16)$ *OR DIFS* $= \{26\text{-}40\}$

our analysis shown for extreme and moderate CW cheating. The results shown in Figures 6-8 are each from single independent trials.

Figure 6 shows that the attack model's foundation is such that since the attacker steals irrespective of the number of nodes at any given instance of time, we should be able to see the difference in distributions. This result justifies our anomaly based detection approach. It also illustrates our technique's stability in larger networks.

Secondly, we simulated a group of attackers to check the robustness of our detection scheme against a group of colluding attackers. As seen in Figure 7, when the attackers perform the same changes to the protocol, the technique should see a cluster of good and bad nodes.

Similarly, the distance between attacker and legitimate delay distributions persists independent of the transport protocol, as shown in Figure 8. Figure 8a shows saturated constant rate transmissions. Figure 8b shows exponential flows with varying frequency of transmissions (that is, with varying ON/OFF periods). This is an important result as it shows that our technique supports different higher layer behavior. It is important to note that the *closeness* between the TCP attacker and UDP legitimate traffic is minimal, but that becomes a trivial concern if traces from a traffic class are compared with those from the same class. It is possible to extract the transport protocol from the IP header of incoming traffic flows even if the data is encrypted because the detection is performed on the wired side. Additionally, we could observe TCP ACKs to deem a flow as TCP.

Also, as one of the scenarios to test the system, we introduced mobility in the nodes and varied their locations. Different combinations of node placements were tried within a given topology. The attacker could be placed close to or far from the other nodes. Alternatively, he could be close to a few and far from the others. Also, he could be close to (worst-case misbehavior) or far from (best-case misbehavior) the base station. The idea behind this type of testing is to see how the *closeness* varies for different scenarios. As will be shown in the next section, there is minimal overall variance in accuracy.

Our scheme is free of adaptive cheating because the bad node's influence over good nodes invariably shows up in a comparison of distributions. Since our technique does not *expect* specific parameters, it is free of parametric adaptive cheating where a clever attacker may choose a different parameter from the one being monitored for. Also, since we do not look at magnitude based metrics, we do not suffer from effective-mean based adaptive cheating, where an attacker

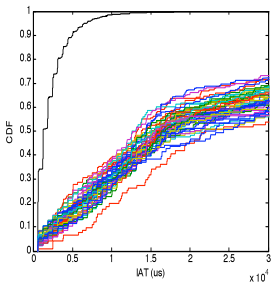

Fig. 6. Larger network.

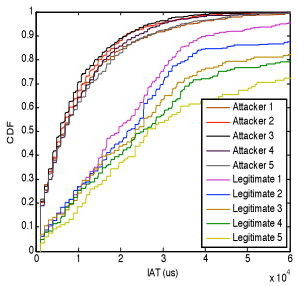

Fig. 7. Colluding attackers.

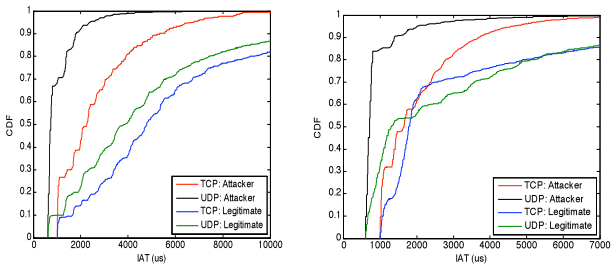

Fig. 8. UDP vs. TCP – (a) Saturated, (b) Bursty.

adjusts his delay parameters in timely intervals to misbehave, while at the same time satisfying the expected mean criteria. Having presented qualitative analysis of the general idea of the technique in this section, we illustrate the accuracy of our scheme in the next section. Note that the scenarios outlined in this section were used in measuring the system's accuracy presented in the next section.

VI. Accuracy Measures

A combination of simulation and experimentation was performed to validate the classifier. The simulation setup used is as discussed in the previous section. The experimental testbed used for validating the technique is as described in Section III-*B*. The laptops act as clients sending data to the desktop. One of the laptops is the attacker, while the other laptops are legitimate nodes.

Simulations are used to evaluate DIFS cheating detection. Also, for the sake of accuracy in emulating physical layer rate adaptation, simulations are used to evaluate rate adaptation cheating detection.

Simulations and experiments are used to evaluate CW cheating detection. For experimentation, CW cheating is performed by modifying the CW values in the *madwifi* [18] driver. A simple method for doing this is by taking advantage of the wireless QoS command line provisioning in 802.11*e* compatible wireless routers. In such a setting, it is possible to configure the Enhanced Distributed Coordination Access (EDCA) parameters (including CW) for each QoS traffic class - best effort, background, voice and video.

A. Misbehavior Detection

As a preliminary measure towards testing the precision of detection, the width of the bins used in the Bayesian approach

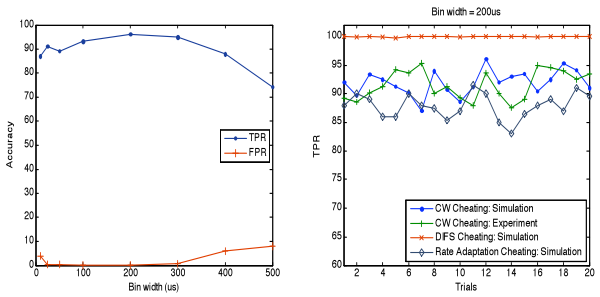

Fig. 9. DCF and Rate adaptation Cheating:
(a) Classifier Bin width tuning, (b) Classifier Accuracy.

was tuned in an effort to determine the optimal width - one that yields peak accuracy. We assume equal-sized bins and perform a non-parametric density estimation. Hence, online computation of bin size via *optimal binning* [19] is not required. Note that most of the histograms produced or published have equal-width bins [20], which justifies our assumption.

Traces from simulations and experiments were used in evaluating accuracy, which applies across all three types of cheating. Traces from 60 windows (20 for each type of cheating) of 10 seconds each were fed into the classifier. The detection from the 60 trials was used in determining True Positive Ratio (TPR) measures for the classifier. Further, 20 trials of legitimate traffic were used to evaluate the False Positive Ratio (FPR). This procedure was followed for different bin widths. Results are shown in Figure 9a.

The accuracy measures shown in Figure 9a are an average of the results from detection trials of all three types of cheating. In Figure 9a, note that with an increase in bin width the accuracy drops, which makes sense as the classifier works better with a higher number of bins.

An optimal bin width of 200μs was chosen, as it gives the minimum FPR of 0.1 and maximum TPR of 96. The resulting measurement is deemed the accuracy of the system because it maximizes TPR and minimizes FPR.

On testing the system with the chosen parameters (bin width = 200*us* and FPR = 0.1) for a total of 20 additional trials (for each cheating type), it was observed that the technique is accurate in detection 100% of the time for DIFS cheating, approximately 92% of the time for CW cheating, and approximately 86% of the time for rate adaptation cheating - as shown in Figure 9b. For the CW cheating trials in Figure 9b, both simulations and experiments were performed (20 trials each).

In the simulations for all three types of cheating, traces from different scenarios were tested, in each of which network parameters were changed (as discussed in the previous section) in an attempt to test the scheme's robustness.

VII. Conclusion and Future Work

We presented a scalable and immediately deployable solution to the problem of selfish behavior at the MAC layer. MAC misbehavior is especially dangerous as it can easily lead to denial of service. Since we do not look for a specific kind of behavior, and instead look for abnormal activity, the system can be extended to detect any variant of cheating, including requesting for a larger duration and resulting in larger NAV values being set by other nodes. Our method is simple and can be easily implemented on the wired side. It does not require changes to be made to the protocol or wireless terminals. Our model is free of adaptive cheating and is independent of number of nodes in the network, protocol, rate adaptation, higher layer behavior, etc. Also, our method works independent of whether RTS/CTS is enabled or not.

As an extension, we plan to consider forms of misbehavior not addressed in this paper. Additionally, we will investigate misbehavior in ad hoc networks.

References

[1] A.L. Toledo, and X. Wang, "Robust Detection of Selfish Misbehavior in Wireless Networks," IEEE Journal on Selected Areas in Communications, Vol.25, 2007.
[2] Y. Rong, S.-K. Lee, and H.-A. Choi, "Detecting Stations Cheating on Backoff Rules in 802.11 Networks Using Sequential Analysis," *IEEE INFOCOM* 2006.
[3] L. Guang, and C. Assi, "Mitigating Smart Selfish MAC Layer Misbehavior in Ad Hoc Networks," *Wireless and Mobile Computing, Networking and Communications,* 2006.
[4] P. Kyasanur, and N.H. Vaidya, "Selfish MAC layer misbehavior in wireless networks," *IEEE Transactions on Mobile Computing,* Vol.4, 2005.
[5] M. Raya, I. Aad, J.-P. Hubaux, and A El Fawal, "DOMINO: Detecting MAC Layer Greedy Behavior in IEEE 802.11 Hotspots," *IEEE Transactions on Mobile Computing,* Vol.5, 2006.
[6] S. Radosavac, G.V. Moustakides, J.S. Baras, and I. Koutsopoulos, "An Analytic Framework for Modeling and Detecting Access Layer Misbehavior in Wireless Networks," *ACM TISSEC*, Vol.11, 2008.
[7] S. Radosavac, A.A. Cárdenas, J.S. Baras, and G.V. Moustakides, "Detecting IEEE 802.11 MAC layer misbehavior in ad hoc networks: Robust strategies against individual and colluding attackers," *Journal of Computer. Security,* Vol.15, 2007.
[8] V.N. Lolla, L.K. Law, S.V. Krishnamurthy, C. Ravishankar, and D. Manjunath, "Detecting MAC Layer Back-off Timer Violations in Mobile Ad Hoc Networks," *IEEE ICDCS* 2006.
[9] M. Cagalj, S. Ganeriwal, I. Aad, and J.-P. Hubaux, "On selfish behavior in CSMA/CA networks," *IEEE INFOCOM* 2005.
[10] J. Youngmi, and G. Kesidis, "Distributed Contention Window Control for Selfish Users in IEEE 802.11 Wireless LANs," *IEEE Journal on Selected Areas in Communications*, Vol.25, 2007.
[11] L. Guang, and C. Assi, "A Self-Adaptive Detection System for MAC Misbehavior in Ad Hoc Networks," *IEEE ICC* 2006.
[12] A. Kamerman, and L. Monteban, "WaveLAN-II: A high-performance wireless LAN for the unlicensed band," *Bell Labs Technical Journal,* pages 118-133, 1997.
[13] Q. Pang, V.C.M Leung, and S.C. Liew, "A rate adaptation algorithm for IEEE 802.11 WLANs based on MAC-layer loss differentiation," *Broadband Networks,* 2005.
[14] J. Kim, S. Kim, S. Choi, and D. Qiao, "CARA: Collision-Aware Rate Adaptation for IEEE 802.11 WLANs," *IEEE INFOCOM* 2006.
[15] S. Biaz, and S. Wu, "Loss Differentiated Rate Adaptation in Wireless Networks," *IEEE WCNC* 2008.
[16] S. H. Wong, H. Yang, S. Lu, and V. Bharghavan, "Robust rate adaptation for 802.11 wireless networks," *ACM MobiCom* 2006.
[17] http://www.isi.edu/nsnam/ns/
[18] http://madwifi-project.org/
[19] M. Refaat, "Data Preparation for Data Mining Using SAS", *Academic Press*, 2007.
[20] http://www.stata.com/support/faqs/graphics/histvary.html