Practical Node Policing in 802.11 WLANs

Hessan Feghhi, Paul Patras and David Malone
Hamilton Institute, National University of Ireland Maynooth
Maynooth, Co. Kildare, Ireland
E-mail: {hessan.feghhi,paul.patras,david.malone}@nuim.ie

***Abstract*—As open-source WiFi device drivers are increasingly available, wireless equipment can be configured to disobey the 802.11 specification, with the goal of achieving performance gains, to the detriment of fair users. We demonstrate a practical implementation of a node policing scheme that combats such selfish behaviour, using commercial off-the-shelf hardware and a modified firmware. With a small testbed, we show that access points running our scheme can detect misbehaving stations, inflict punishment upon them and effectively restore fairness in the network.**

I. Introduction

Channel access in current IEEE 802.11 wireless LANs (WLANs) is usually controlled by the Distributed Coordination Function (DCF) [1]. Specifically, stations run independent DCF instances, which assign transmission opportunities in a decentralised fashion. Thus, each node is responsible for its own behaviour and there is little incentive to play by the standard rules if changes can be made to DCF. In fact, open-source device drivers (e.g. MadWifi,[1] compat-wireless,[2] etc.) provide flexible access to the Medium Access Control (MAC) layer primitives, which enable users with minimal programming skills to reconfigure their interfaces and boost their performance, while starving compliant contenders.

To understand the impact of such attacks, consider a very simple case of a real network with a regular access point (AP) serving two clients, both sending backlogged UDP traffic with payload size of 1000 Bytes, at 11 Mb/s PHY rate. One of them employs compliant channel access, and the other competes with a Contention Window minimum parameter half the size of the standard's default value (i.e. $CW_{min} = CW_{min}^{default}/2$). Note that with e.g. Atheros chipset based wireless cards and Linux operating system, this is achievable by simply invoking an `iwpriv` system command. As depicted in Fig. 1 (with light bars), in this scenario the cheating node attains a throughput figure more than twice that of the fair station, while the performance of the other is clearly affected. We note that more sophisticated attacks that manipulate the arbitration inter-frame spacing (AIFS) or the transmission opportunity limit (TXOP) parameters would have even more significant impact on the compliant users.

[1]`http://www.madwifi-project.org`
[2]`http://wireless.kernel.org/en/users/Drivers`

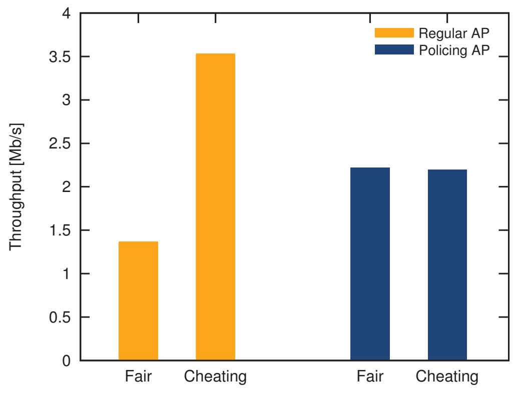

Figure 1. Comparison of throughput performance with one attacker.

To address this problem, we have implemented a scheme to police an 802.11 WLAN, whereby the AP measures each station's performance and then controls a probability P_{ack} that a node will not be sent an acknowledgement (ACK) after a successful transmission, thus forcing it to backoff and double its contention window, thereby reducing its transmission probability. Considering the earlier example scenario, this approach can successfully restore throughput fairness between compliant and misbehaving nodes, as shown in Fig. 1 (with dark bars).[3] Our technique builds upon earlier work [2], where analysis and simulation foresee an appropriate controller can effectively tackle MAC misbehaviour. However, implementing ACK suppression with real devices is a challenging task, since automatic repeat-request (ARQ) is a basic operation handled at a low level of the network stack. We tackle this challenge by exploiting the OpenFWWF firmware,[4] which offers high flexibility, including access to functionality such as ACK generation, and develop a practical policing implementation that we deploy with commodity hardware.

To demonstrate the effectiveness of our prototype, we showcase a real scenario where a misbehaving user, engaged in a file transfer operation, manipulates their MAC configuration to reduce transfer duration, while jeopardizing the

[3]We disable rate adaptation, but generally assume nodes run state-of-the-art rate control algorithms, e.g. Minstrel, which only act upon failures due to channel errors.
[4]`http://www.ing.unibs.it/~openfwwf/`

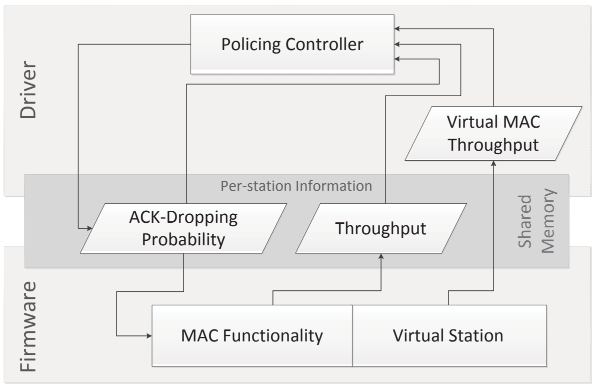

Figure 2. Policing implementation.

performance of a competing video flow. Using a Linux-based AP equipped with a commercial off-the-shelf Broadcom WiFi card, we show that our implementation detects the cheater, rapidly counteracts the observed behaviour and restores the quality of the video stream.

II. Policing Algorithm

The operation of the policing algorithm relies on the fundamental nature of ACK reception at the client side. In our scheme, when the AP decides not to generate an ACK, it also drops the received frame, just as if it was corrupt. Consequently, even misbehaving stations must resend the packet to ensure successful delivery. Further, the subsequent retries occur less frequent, given the CW doubling. Otherwise, aggressive nodes employing fixed CW settings will be eventually disassociated. This technique only requires changes at the AP and does not change the 802.11 rules. Further, it adds no additional signalling between AP and well-behaved nodes for detecting cheaters, which makes it easy to deploy.

P_{ack} **controller operation:** The algorithm is executed iteratively and involves controlling, at each step, the probability $P_{ack,i}$, with which the frames of a station i will not be acknowledged during the next cycle [2]:

$$P^{k+1}_{ack,i} = \left[P^k_{ack,i} + \alpha \left(\frac{S^k_i}{S^k_f} - (1 - \gamma P^k_{ack,i}) \right) \right]_{0,1},$$

where $[x]_{a,b} = \max(a, \min(b, x))$, S^k_i is the throughput of client i, S^k_f is the maximum rate a well-behaved node can attain, $\alpha > 0$ is a parameter controlling adaptation rate and $\gamma > 0$ introduces a penalty for misbehaviour.

Fair throughput estimation: Unlike previous works that focus on identifying the mechanisms of cheating and counteract (if at all) depending on their nature (e.g. [3], [4]), with our approach the AP estimates the maximum throughput of a fair station S_f given the current conditions and remains agnostic to the form of misbehaviour. Specifically, we observe the channel states (idle/busy) at the AP to estimate the collision probability p seen by a "virtual" fair MAC instance that follows the 802.11 protocol but does not actually inject any traffic, which we compute as follows:

$$p = \frac{\#\text{busy slots}}{\#\text{busy slots} + \#\text{idle slots}}.$$

Subsequently, we use this do derive the expected "fair" throughput performance S_f using a model [5].

Implementation: We implemented our mechanisms on x86-based computers, equipped with Broadcom B4318 adapters, running Debian Linux (kernel 2.6.32) and the `mac80211` framework with modified `b43` driver and OpenFWWF firmware. Fig. 2 depicts the key building blocks.

The ACK dropping resides in the firmware, as this is a time sensitive task. A table with P_{ack} values corresponding to each associated station is stored in the shared memory. Upon a frame reception, a look-up routine fetches the appropriate P_{ack} and then skips ACK generation with the appropriate probability. The virtual MAC is also included in the firmware and runs whenever the adapter processing unit is in idle state. Note that this happens not only when the channel is idle, but also when transmissions/receptions are ongoing, thus we obtain estimates with good accuracy.

ACK drop probabilities must be updated periodically, to account for the network dynamics. These updates are performed by the driver, which has access to the inputs required and typically runs on a higher performance CPU than the firmware. The P_{ack} updates are performed by measuring per-station throughput and comparing with the estimated fair value.

III. Demo Scenario

To demonstrate the effectiveness of our implementation, we consider the scenario in Fig. 3, where station 1 is

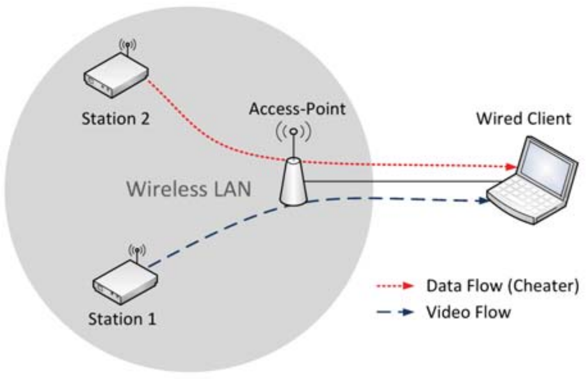

Figure 3. Demo scenario.

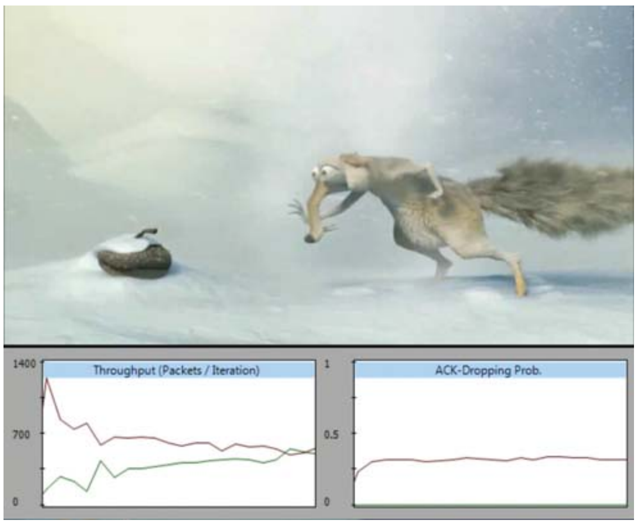

Figure 4. Demo visualisation tools: the received video stream is displayed at the top of the GUI; the plots on the bottom show in real time the measured throughput (left) and the applied ACK dropping probabilities (right).

streaming a 2 Mb/s MPEG-2 encoded video of "Ice Age 3: Dawn of the dinosaurs" teaser, with an approximate duration of 2 minutes. Simultaneously, station 2 is uploading a large data file to the wired network. Nodes are configured with the HR/DSSS (802.11b) PHY layer parameters (11 Mb/s data rate). The policing algorithm runs at the AP and a set of visualisation tools are deployed on the wired client, both to illustrate in real-time the operation of the policing scheme (i.e. the time evolution of P_{ack} and the measured per-station throughput S_i), and to permit the playback and subjective evaluation of the quality of the video (see Fig. 4).

Normal operation: First, we consider that both stations follow the IEEE 802.11 standard rules and show that under such circumstances the video is streamed with no performance degradation, while the remaining channel capacity is used to accommodate the transfer of the data file. We use this setup as a benchmark for comparing the throughput and video performance when the AP operates with and without the policing scheme, and the node sending data is cheating, as we explain next.

Node misbehaviour, without policing: To show the impact of MAC misbehaviour, we configure the CW_{min} parameter of station 2 to half of the value recommended by the standard and repeat the previous experiment. This setting increases the aggressiveness of station 2, which will be backlogged with packets. Consequently, it will consume most of the available capacity, thereby leaving station 1 unable to satisfy the bandwidth demands of the video flow. Using our graphical interface, we show that the cheater achieves a throughput value nearly twice that of the fair node, while the degradation of the video quality can be easily observed from the frequently garbled playback.

Node misbehaviour, with policing: Finally, we activate the node policing mechanism at the AP and repeat the experiment with one cheating node sending data and a fair station streaming video. We show that the policing algorithm quickly detects the attack and increases the ACK dropping probability corresponding to the misbehaving node. As a result, station 2 is forced to backoff periodically, which effectively reduces its transmission attempt rate and consequently limits the attained performance. This is visible in the time evolution of the throughput displayed in the GUI, which is immediately throttled down to the fair value, leaving more transmission opportunities for the fair contender. Conversely, as station 1 is following the standard specifications, its corresponding P_{ack} probability remains effectively at zero, and the quality of the transmitted video is preserved.

IV. Conclusion

Given the increased number of wireless devices that permit modifying the 802.11 MAC parameters, users have little incentive to conform to the standard specification. With this demo, we showcase a practical solution that can be implemented with commodity hardware and effectively polices 802.11 misbehavoiur, without requiring any cooperation between the AP and clients, therefore being fully standard compliant.

Acknowledgements

This work was supported by the European Commission (FP7 grant agreement n. 257263 – FLAVIA project) and Science Foundation Ireland (grant 08/SRC/I1403).

References

[1] IEEE 802.11 WG, *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications*. IEEE 802.11 Standard, 2007.

[2] I. Dangerfield, D. Malone, and D. Leith, "Incentivising Fairness and Policing Nodes in WiFi," *IEEE Communications Letters*, vol. 15, pp. 500–50, May 2011.

[3] P. Serrano, A. Banchs, V. Targon, and J. F. Kukielka, "Detecting selfish configurations in 802.11 WLANs," *IEEE Communications Letters*, vol. 14, no. 2, Feb. 2010.

[4] M. Raya, I. Aad, J.-P. Hubaux, and A. E. Fawal, "DOMINO: Detecting MAC layer greedy behavior in IEEE 802.11 hotspots," *IEEE Transactions on Mobile Computing*, vol. 5, pp. 1691–1705, Dec. 2006.

[5] G. Bianchi, "Performance analysis of IEEE 802.11 distributed coordination function," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 3, Mar. 2000.