Computer Communications 110 (2017) 175–186

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

μNap: Practical micro-sleeps for 802.11 WLANs

Arturo Azcorra[a,b], Iñaki Ucar[a,*], Francesco Gringoli[c], Albert Banchs[a,b], Pablo Serrano[a]

[a] Universidad Carlos III de Madrid, Leganés, 28911 Spain
[b] IMDEA Networks Institute, Leganés, 28918, Spain
[c] Universitá degli Studi di Brescia, Brescia, 25123 Italy

ARTICLE INFO

Article history:
Received 11 November 2016
Revised 30 May 2017
Accepted 23 June 2017
Available online 29 June 2017

Keywords:
Energy efficiency
Energy measurement
CSMA wireless networks

ABSTRACT

In this paper, we revisit the idea of putting interfaces to sleep during *packet overhearing* (i.e., when there are ongoing transmissions addressed to other stations) from a practical standpoint. To this aim, we perform a robust experimental characterisation of the timing and consumption behaviour of a commercial 802.11 card. We design μNap, a local standard-compliant energy-saving mechanism that leverages micro-sleep opportunities inherent to the CSMA operation of 802.11 WLANs. This mechanism is backwards compatible and incrementally deployable, and takes into account the timing limitations of existing hardware, as well as practical CSMA-related issues (e.g., capture effect). According to the performance assessment carried out through trace-based simulation, the use of our scheme would result in a 57% reduction in the time spent in overhearing, thus leading to an energy saving of 15.8% of the activity time.

1. Introduction

IEEE 802.11 is the standard *de facto* for broadband Internet access. The recent 802.11ac amendment opens up new opportunities by bringing Gigabit to wireless local area networks (WLANs). Since the seminal work [1], energy efficiency stands as a major issue due to the intrinsic CSMA mechanism, which forces the network card to stay active performing *idle listening*.

The 802.11 standard developers are fully aware of the energy issues that WiFi poses on battery-powered devices and have designed mechanisms to reduce energy consumption. One of such mechanisms is the Power Save (PS) mode, which is widely deployed among commercial wireless cards, although unevenly supported in software drivers. With this mechanism, a station (STA) may enter a doze state during long periods of time, subject to prior notification, if it has nothing to transmit. Meanwhile, packets addressed to this dozing STA are buffered and signalled in the Traffic Indication Map (TIM) attached to each beacon frame.

The PS mechanism dramatically reduces the power consumption of a wireless card. However, the counterpart is that, since the card is put to sleep for hundreds of milliseconds, the user experiences a serious performance degradation because of the delays incurred. The automatic power save delivery (APSD) introduced by the 802.11e amendment (Perez-Costa and Camps-Mur [2] gives a nice overview) is based on this mechanism, and aims to improve the downlink delivery by taking advantage of QoS mechanisms, but has not been widely adopted.

More recently, the 802.11ac amendment improves the PS capabilities with the VHT TXOP power save mechanism. Basically, an 11ac STA can doze during a TXOP (transmission opportunity) in which it is not involved. This capability is announced within the new VHT (Very High Throughput) framing format, so that the AP knows that it cannot send traffic to those STAs until the TXOP's natural end, even if it is interrupted earlier. Still, the potential dozing is in the range of milliseconds and may lead to channel underuse if these TXOPs are not fully exploited.

Considering shorter timescales, packet overhearing (i.e., listening to the wireless while there is an ongoing transmission addressed to other station) has been identified as a potential source of energy wastage [3]. Despite this, we have performed an extensive measurement campaign and have not found any attempt from manufacturers[1] to implement solutions to lessen its impact.

In this work, we revisit this idea of packet overhearing as a trigger for sleep opportunities, and we take it one step further to the range of microseconds. To this end, we experimentally explore the timing limitations of 802.11 cards and, building on this knowledge,

* Corresponding author.
E-mail addresses: azcorra@it.uc3m.es (A. Azcorra), inaki.ucar@uc3m.es (I. Ucar), francesco.gringoli@unibs.it (F. Gringoli), banchs@it.uc3m.es (A. Banchs), pablo@it.uc3m.es (P. Serrano).

http://dx.doi.org/10.1016/j.comcom.2017.06.008
0140-3664/

[1] Using our setup described in Section 3, we have tested cards from different manufacturers with the latest available firmwares and drivers: Broadcom BCM43224, Realtek RTL8191SEvB, Atheros AR9280, Intel Wireless-AC 7260 and Qualcomm Atheros QCA9880, which is a very recent state-of-the-art 11ac card.

we design μNap, a local standard-compliant energy-saving mechanism for 802.11 WLANs. With μNap, a STA is capable of saving energy during packet overhearing autonomously, with full independence from the 802.11 capabilities supported or other power saving mechanisms in use, which makes it backwards compatible and incrementally deployable.

In summary, the main contributions of this paper are the following:

- A robust experimental characterisation of the timing and consumption behaviour of a COTS (commercial off-the-shelf) wireless card.
- Design of μNap, a local standard-compliant energy-saving mechanism that takes into account these timing limitations, as well as practical CSMA-related issues (e.g., capture effect, hidden nodes) not considered in prior work.
- Performance evaluation of μNap based on our measurements and real wireless traces, and performance degradation analysis due to channel errors.
- Discussion of the impact and applicability of our mechanism. We draw attention to the need for standardising the hardware capabilities in terms of energy in 802.11.

The remainder of this paper is organised as follows. Section 2 reviews related work. Section 3 experimentally explores the timing limitations of 802.11 COTS devices. Section 4 analyses micro-sleep opportunities in CSMA as well as several practical issues of 802.11 networks, and proposes μNap. Section 5 presents the performance evaluation and Section 6 summarises the conclusions of this paper.

2. Related work

It has been empirically proved that, in any network, the so-called power-law distribution, also known as Pareto distribution, holds for the traffic generated by its nodes. In other words, typically a few heavy hitters-generate most of the traffic while the majority of the nodes are only responsible for a small fraction, and this is true regardless of the network load. This means that the majority of nodes within any WLAN would spend most of the time in idle state.

There are two main strategies to save energy in this idle time: the first one targets *idle listening* (the wireless channel is empty), and the second one targets *packet overhearing* (there are other nodes communicating). To support these savings, COTS devices have two main operational states as a function of the reference clock used: the active state and the sleep state. The more a card stays in sleep state, the less power it consumes.

Since its conception, 802.11 has attempted to minimise idle listening with the introduction of the PS mode, and some previous work followed this path. For instance, Liu and Zhong [4] proposed μPM to exploit short idle intervals ($<$100 ms) without buffering or cooperation. μPM predicts the arrival time of the next frame and puts the interface in PS mode while no arrivals are expected. This mechanism demonstrated poor granularity (tens of ms) on existing hardware and leads to performance degradation due to frame loss. Therefore, it is only suitable for low-traffic scenarios.

Others propose a PS-like operation. Jang et al. [5] described Snooze, an access point (AP)-directed micro-sleep scheduling and antenna configuration management method for 11n WLANs. As a consequence of its centralised design, the granularity of the so-called micro-sleeps in this approach is poor (few milliseconds), which poses doubts on its performance under heavy loads.

Zhang and Shin [6] addressed the issue from a different standpoint with their Energy-Minimizing Idle Listening (E-MiLi). E-MiLi adaptively downclocks the card during idle periods, and reverts to full rate when an incoming frame is detected. To achieve this purpose, they need to change the physical layer (PHY) all the way down to enable downclocked detection, which severely limits the potential gains. For instance, the E-MiLi downclocking factor of 16 would yield a high power consumption in a modern card compared to its sleep state (see Section A.2).

On the other hand, all indicators show that we should expect an exponential grow in the number of wireless devices connected. Thus, there is a rough consensus about that *densification* will become one of the main aspects of next-generation wireless networks, which brings us back to the problem of packet overhearing. In this way, the recent 11ac amendment adds the ability to save energy during TXOPs, but this mechanism is restricted to QoS traffic, and the potential sleeps are coarse, in the range of milliseconds. Any sub-millisecond approach must take into account the timing parameters of the hardware. In fact, some early studies realise the importance of this issue when WiFi technology began to take off commercially [7–9].

Baiamonte and Chiasserini [10] were the first to chase fine-grained micro-sleep opportunities during packet overhearing. They define the Energy-efficient Distributed Access (EDA) scheme, which uses the 802.11 virtual carrier-sensing mechanism for power-saving purposes. Basically, a STA dozes when the Network Allocation Vector (NAV) or the backoff counter are non-zero. Unfortunately, this work lacks an empirical characterisation of the timing constraints needed to design a practical mechanism. Moreover, dozing during the backoff window is not 802.11-fair: in 802.11, STAs must sense the channel every single time slot during the contention period and, if another STA seizes the channel first, the backoff timer must be stopped in order to receive the incoming frame and set the NAV to the proper value. The EDA scheme allows STAs to doze during the contention period and, therefore, breaks the CSMA operation.

Balaji et al. [11] revisited the problem of packet overhearing with a scheme called Sleep during Neighbor-Addressed Frame (SNAF). With SNAF, a wireless card checks the destination MAC address and switches to sleep state during the payload duration if it was addressed to other host. They assume, without any experimental validation though, an instantaneous switch-off and that the time required to wake up is equivalent to a Short Interframe Space (SIFS). In order to prevent the risk resulting from errors in the frame header that would lead to an incorrect NAV counter, the authors propose to introduce a new framing format with a new FCS devoted to the MAC header only. This solution lacks compatibility and introduces more overhead based on no evidence.

Building on the same idea, Sudarshan et al. [12] proposed bersleep. This time, the authors do not consider it necessary to add any extra FCS, as they claim (without any specific basis) that such errors are very unlikely.

More recently, Palacios-Trujillo et al. modified DCF [13] and PCF [14] to exploit per-packet sleeps. They also applied these ideas to network coding [15] and to a polling-based version of 11ac's TXOP PS mode [16]. Unfortunately, all these papers rely on these early studies mentioned before [7–9], which analysed old wireless cards unable to perform sub-millisecond transitions between states.

3. State transition times on 802.11 cards

From the hardware point of view, the standard PS mechanism requires supporting two states of operation: the *awake state* and the *sleep state*. The latter is implemented using a secondary low-frequency clock. Indeed, it is well-known that the power consumption of digital devices is proportional to the clock rate [6]. In fact, other types of devices, such as microcontroller-based devices or modern general-purpose CPUs, implement sleep states in the same way.

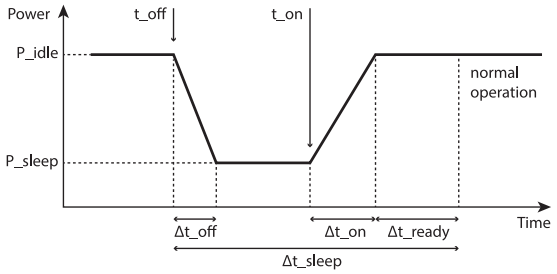

Fig. 1. Generic sleep breakdown.

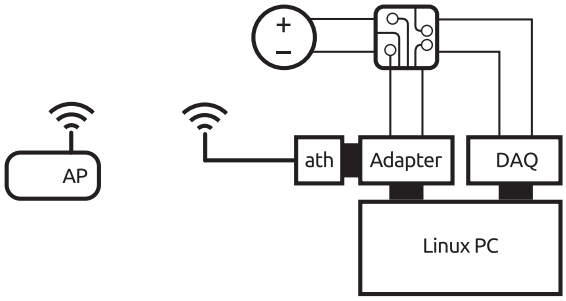

Fig. 2. Measurement setup for energy performance characterisation.

For any microcontroller-based device with at least an idle state and a sleep state, one would expect the following behaviour for an ideal sleep. The device was in idle state, consuming P_{idle}, when, at an instant t_{off}, the sleep state is triggered and the consumption falls to P_{sleep}. A secondary low-power clock decrements a timer of duration $\Delta t_{\text{sleep}} = t_{\text{on}} - t_{\text{off}}$, and then the expiration of this timer triggers the wake-up at t_{on}. The switching between states would be instantaneous and the energy saving would be

$$E_{\text{save}} = (P_{\text{idle}} - P_{\text{sleep}}) \cdot \Delta t_{\text{sleep}} \tag{1}$$

This estimate could be considered valid for a time scale in the range of tens of milliseconds at least, but this is no longer true for micro-sleeps. Instead, Fig. 1 presents a conceptual breakdown of a generic micro-sleep. After the sleep state is triggered at t_{off}, it takes Δt_{off} before the power consumption actually reaches P_{sleep}. Similarly, after the wake-up is triggered at t_{on}, it takes some time, Δt_{on}, to reach P_{idle}. Finally, the circuitry might need some additional time Δt_{ready} to stabilise and operate normally. Thus, the most general expression for the energy saved in a micro-sleep is the following:

$$\begin{aligned} E'_{\text{save}} &= E_{\text{save}} - E_{\text{waste}} \\ &= (P_{\text{idle}} - P_{\text{sleep}}) \cdot (\Delta t_{\text{sleep}} - \Delta t_{\text{ready}}) \\ &\quad - \int_{\Delta t_{\text{off}} \cup \Delta t_{\text{on}}} (P - P_{\text{sleep}}) \cdot dt \end{aligned} \tag{2}$$

where we have considered a general waveform $P(t)$ for the transients Δt_{off} and Δt_{on}. E_{waste} represents an energy toll per sleep when compared to the ideal case.

Our next objective is to quantify these limiting parameters, which can be defined as follows:

- Δt_{off} is the time required to switch from idle power and to sleep power consumption.
- Δt_{on} is the time required to switch from sleep power to idle power consumption.
- Δt_{ready} is the time required for the electronics to stabilise and become ready to transmit/receive.

The sum of this set of parameters defines the minimum sleep time, $\Delta t_{\text{sleep, min}}$, for a given device:

$$\Delta t_{\text{sleep,min}} = \Delta t_{\text{off}} + \Delta t_{\text{on}} + \Delta t_{\text{ready}} \tag{3}$$

Performing this experimental characterisation requires the ability to timely trigger the sleep mode on demand. Most COTS cards are not suitable for this task, because they implement all the low-level operations in an internal proprietary binary firmware. After an extensive study, we found that cards based on the open-source driver `ath9k` are well suited for our needs, as they do not load a firmware to operate, and the driver has access to very low level functionality (e.g., supporting triggering the sleep mode by just writing into a register). Leveraging on these properties, we conducted our experimental characterisation using an Atheros AR9280 Half-height Mini PCI Express card.

This card is attached to a PC through a flexible x1 PCI Express to Mini PCI Express adapter from Amfeltec, as the right part of Fig. 2 depicts. This adapter connects the PCI bus' data channels to the host and provides an ATX port so that the wireless card can be supplied by an external power source. The same PC holds a NI PCI-6289, a high-accuracy multifunction data acquisition (DAQ) device, optimised for 18-bit input accuracy. Its timing resolution is 50 ns with an accuracy of 50 ppm of sample rate. In this way, the operations sent to the wireless card and the energy measurements can be correlated using the same timebase. A small command-line tool was developed[2] to perform measurements on the DAQ card using the open-source Comedi[3] drivers and libraries.

The power supply is a Keithley 2304A DC Power Supply, which is optimised for testing battery-operated wireless communication devices. It powers the wireless card through a measurement circuit that extracts the voltage and converts the current with a high-precision sensing resistor and amplifier. Considering that the DAQ card has a certain settling time, it can be modelled as a small capacitor which acts as a low-pass filter. Thus, two buffers (voltage followers) are placed before the DAQ card to decrease the output impedance of the measurement circuit [17].

The card under test is associated to an AP in 11a mode to avoid any interfering traffic from neighbouring networks. This AP is placed very close to the node to obtain the best possible signal quality, as we are simply interested in not losing the connectivity for this experiment. With this setup, the idea is to trigger the sleep state, then bring the interface back to idle and finally trigger the transmission of a buffered packet as fast as possible, in order to find the timing constraints imposed by the hardware in the power signature. From an initial stable power level, with the interface associated and in idle mode, we would expect a falling edge to a lower power level corresponding to the sleep state. Then the power level would raise again to the idle level and, finally, a big power peak would mark the transmission of the packet. By correlating the timestamps of our commands and the timestamps of the measured power signature, we are able to measure the limiting parameters Δt_{off}, Δt_{on}, Δt_{ready}.

The methodology to reproduce these steps required hacking the `ath9k` driver to timely trigger write operations in the proper card registers, and to induce a transmission of a pre-buffered packet directly in the device without going through the entire network stack. The code for reproducing this experiment is available on GitHub[4], and comprises the following steps:

0. Initially, the card is in idle state, connected to the AP.
1. A RAW socket (Linux AF_PACKET socket) is created and a socket buffer is prepared with a fake packet.

[2] https://github.com/Enchufa2/daq-acquire.
[3] http://comedi.org/.
[4] https://github.com/Enchufa2/crap/tree/master/ath9k/downup.

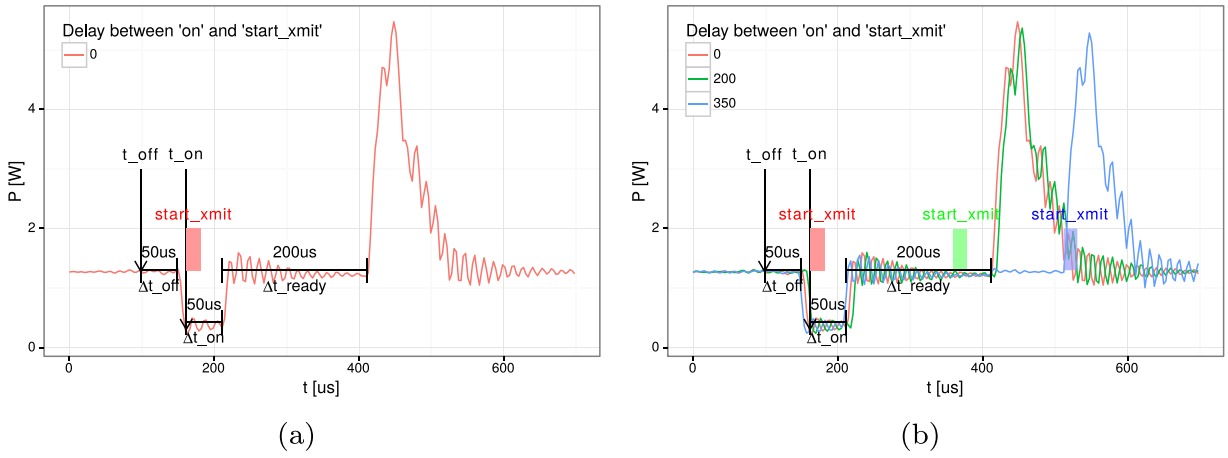

Fig. 3. Atheros AR9280 timing characterisation.

2. t_{off} is triggered by writing a register in the card, which has proved to be almost instantaneous in kernel space.
3. A micro-delay of 60 µs is introduced in order to give the card time to react.
4. t_{on} is triggered with another register write.
5. Another timer sets a programmable delay.
6. The fake frame is sent using a low-level interface, i.e., calling the function `ndo_start_xmit()` from the `net_device` operations directly. By doing this, we try to spend very little time in kernel.

The power signature recorded as a result of this experiment is shown in Fig. 3(a). As we can see, the card spends $\Delta t_{\text{off}} = 50$ µs consuming P_{idle} and then it switches off to P_{sleep} in only 10 µs. Then, t_{on} is triggered. Similarly, the card spends $\Delta t_{\text{on}} = 50$ µs consuming P_{sleep} and it wakes up almost instantaneously. Note that the transmission of the packet is triggered right after the t_{on} event and the frame spends very little time at the kernel (the time spent in kernel corresponds to the width of the rectangle labelled as `start_xmit` in the graph). Nonetheless, the card sends the packet 200 µs after returning to idle, even though the frame was ready for transmission much earlier.

To understand the reasons for the delay in the frame transmission observed above, we performed an experiment in which frame transmissions were triggered at different points in time by introducing different delays between the t_{on} and `start_xmit` events. Fig. 3(b) shows that the card starts transmitting always in the same instant whenever the kernel triggers the transmission within the first 250 µs right after the t_{on} event (lines 0 and 200). Otherwise, the card starts transmitting almost instantaneously (line 350). This experiments demonstrate that the device needs $\Delta t_{\text{ready}} = 200$ µs to get ready to transmit/receive after returning to idle.

To sum up, our experiments show that, if we want to bring this card to sleep during a certain time Δt_{sleep}, we should take into account that it requires a minimum sleep time $\Delta t_{\text{sleep,min}} = 300$ µs. Therefore, $\Delta t_{\text{sleep}} \geq \Delta t_{\text{sleep, min}}$ must be satisfied, and we must program the t_{on} interrupt to be triggered $\Delta t_{\text{on}} + \Delta t_{\text{ready}} = 250$ µs before the end of the sleep. Note also that the card wastes a fixed time Δt_{waste} consuming P_{idle}:

$$\Delta t_{\text{waste}} = \Delta t_{\text{off}} + \Delta t_{\text{ready}} \tag{4}$$

which is equal to 250 µs also. Thus, the total time in sleep state is $\Delta t_{\text{sleep}} - \Delta t_{\text{waste}}$, and the energy toll from Eq. (2) can be simplified as follows:

$$E_{\text{waste}} \approx (P_{\text{idle}} - P_{\text{sleep}}) \cdot \Delta t_{\text{waste}} \tag{5}$$

4. μNap design

The key idea of μNap is to put the interface to sleep during packet overhearing while meeting the constraint $\Delta t_{\text{sleep, min}}$ identified in the previous section. Additionally, the algorithm should be local in order to be incrementally deployable, standard-compliant, and should take into account real-world practical issues. For this purpose, Section 4.1 analyses available micro-sleep opportunities in 802.11 and determines under which circumstances the NAV mechanism can be used to extend a micro-sleep while ensuring that no frames are lost within such time. Section 4.2 explores well-known practical issues of WLAN networks that had not been addressed by previous energy-saving schemes. Finally, Section 4.3 presents μNap.

4.1. Micro-sleep opportunities in 802.11

Due to the CSMA mechanism, 802.11 STAs receive every single frame from their SSID or from others in the same channel (even some frames from overlapping channels). Upon receiving a frame, a STA checks the Frame Check Sequence (FCS) for errors and then, and only after having received the entire frame, it discards the frame if it is not the recipient. In 802.11 terminology, this is called *packet overhearing*. Since packet overhearing consumes the power corresponding to a full packet reception that is not intended for the station, it represents a source of inefficiency. Thus, we could avoid this unnecessary power consumption by triggering micro-sleeps that bring the wireless card to a low-energy state.

Indeed, the Physical Layer Convergence Procedure (PLCP) carries the necessary information (rate and length) to know the duration of the PLCP Service Data Unit (PSDU), which consists of a MAC frame or an aggregate of frames. And the first 10 bytes of a MAC frame indicate the intended receiver, so a frame could be discarded very early, and the station could be brought to sleep if the hardware allows for such a short sleeping time. Therefore, the most naive micro-sleep mechanism could determine, given the constraint $\Delta t_{\text{sleep, min}}$, whether the interface could be switched off in a frame-by-frame basis. And additionally, this behaviour can be further improved by leveraging the 802.11 virtual carrier-sensing mechanism.

Virtual carrier-sensing allows STAs not only to seize the channel for a single transmission, but also to signal a longer exchange with another STA. For instance, this exchange can include the acknowledgement sent by the receiver, or multiple frames from a station in a single transmission opportunity (TXOP). So MAC frames carry a duration value that updates the Network Allocation Vector (NAV),

which is a counter indicating how much time the channel will be busy due to the exchange of frames triggered by the current frame. And this duration field is, for our benefit, enclosed in the first 10 bytes of the MAC header too. Therefore, the NAV could be exploited to obtain substantial gains in terms of energy.

In order to unveil potential sleeping opportunities within the different states of operation in 802.11, first of all we review the setting of the NAV. 802.11 comprises two families of channel access methods. Within the legacy methods, the Distributed Coordination Function (DCF) is the basic mechanism with which all STAs contend employing CMSA/CA with binary exponential backoff. In this scheme, the duration value provides single protection: the setting of the NAV value is such that protects up to the end of one frame (data, management) plus any additional overhead (control frames). For instance, this could be the ACK following a data frame or the CTS + data + ACK following an RTS.

When the Point Coordination Function (PCF) is used, time between beacons is rigidly divided into contention and contention-free periods (CP and CFP, respectively). The AP starts the CFP by setting the duration value in the beacon to its maximum value (which is 32 768; Table 8-3 of the IEEE Std 802.11-2012 [18] depicts the duration/ID field encoding). Then, it coordinates the communication by sending CF-Poll frames to each STA. As a consequence, a STA cannot use the NAV to sleep during the CFP, because it must remain CF-pollable, but it still can doze during each individual packet transmission. In the CP, DCF is used.

802.11e introduces traffic categories (TC), the concept of TXOP, and a new family of access methods called Hybrid Coordination Function (HCF), which includes the Enhanced Distributed Channel Access (EDCA) and the HCF Controlled Channel Access (HCCA). These two methods are the QoS-aware versions of DCF and PCF respectively.

Under EDCA, there are two classes of duration values: single protection, as in DCF, and multiple protection, where the NAV protects up to the end of a sequence of frames within the same TXOP. By setting the appropriate TC, any STA may start a TXOP, which is zero for background and best-effort traffic, and of several milliseconds for video and audio traffic as defined in the standard (Table 8-105 of the IEEE Std 802.11-2012 [18]). A non-zero TXOP may be used for dozing, as 11ac does, but these are long sleeps and the AP needs to support this feature, because a TXOP may be truncated at any moment with a CF-End frame, and it must keep buffering any frame directed to any 11ac dozing STA until the NAV set at the start of the TXOP has expired.

HCCA works similarly to PCF, but under HCCA, the CFP can be started at almost any time. In the CFP, when the AP sends a CF-poll to a STA, it sets the NAV of other STAs for an amount equal to the TXOP. Nevertheless, the AP may reclaim the TXOP if it ends too early (e.g., the STA has nothing to transmit) by resetting the NAV of other STAs with another CF-Poll. Again, the NAV cannot be locally exploited to perform energy saving during a CFP.

Finally, there is another special case in which the NAV cannot be exploited either. 802.11g was designed to bring the advantages of 11a to the 2.4 GHz band. In order to interoperate with older 11b deployments, it introduces CTS-to-self frames (also used by more recent amendments such as 11n and 11ac). These are standard CTS frames, transmitted at a legacy rate and not preceded by an RTS, that are sent by a certain STA to itself to seize the channel before sending a data frame. In this case, the other STAs cannot know which will be the destination of the next frame. Therefore, they should not use the duration field of a CTS for dozing.

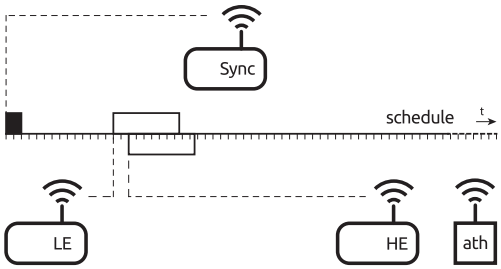

Fig. 4. Measurement setup for the MIM effect.

4.2. Practical issues

4.2.1. Impact of capture effect

It is well-known that a high-power transmission can totally blind another one with a lower SNR. Theoretically, two STAs seizing the channel at the same time yields a collision. However, in practice, if the power ratio is sufficiently high, a wireless card is able to decode the high-power frame without error, thus ignoring the other transmission. This is called *capture effect*, and although not described by the standard, it must be taken into account as it is present in real deployments.

According to [19], there are two types of capture effect depending on the order of the frames: if the high-power frame comes first, it is called first capture effect; otherwise, it is called second capture effect. The first one is equivalent to receiving a frame and some noise after it, and then it has no impact in our analysis. In the second capture effect, the receiving STA stops decoding the PLCP of the low-power frame and switches to another with higher power. If the latter arrives *before* a power-saving mechanism makes the decision to go to sleep, the mechanism introduces no misbehaviour.

However, [19] suggests that a high-power transmission could blind a low-power one at any time, even when the actual data transmission has begun. This is called *Message in Message* (MIM) in the literature [20,21], and it could negatively impact the performance of an interface implementing an energy-efficiency mechanism based on packet overhearing. In the following, we will provide new experimental evidence supporting that this issue still holds in modern wireless cards.

We evaluated the properties of the MIM effect with an experimental setup consisting of a card under test, a brand new 802.11ac three-stream Qualcomm Atheros QCA988x card, and three additional helper nodes. These are equipped with Broadcom KBFG4318 802.11g cards, whose behaviour can be changed with the open-source firmware OpenFWWF [22]. We disable the carrier sensing and back-off mechanisms so that we can decide the departure time of every transmitted frame with 1 µs granularity with respect to the internal 1MHz clock.

Fig. 4 depicts the measurement setup, which consists of a node equipped with our Atheros card under test (*ath*), a synchronization (Sync) node, a *high energy* (HE) node and a *low energy* (LE) node. These two HE and LE nodes were manually carried around at different distances with respect to the *ath* node until we reached the desired power levels.

The Sync node transmits 80-byte long beacon-like frames periodically at 48 Mbps, one beacon every 8192 µs: the time among consecutive beacons is divided in 8 schedules of 1024 µs. Inside each schedule, time is additionally divided into 64 micro-slots of 16 µs. We then program the firmware of the HE and LE nodes to use the beacon-like frames for keeping their clocks synchronised and to transmit a single frame (138-µs long) per schedule starting at a specific micro-slot. This allows us to always start the trans-

Table 1
MIM effect.

ΔP [dB]	Δt [μs]	LE frames		HE frames	
		% rx	% err	% rx	% err
≤ 5	0	0.04	50.00	92.00	17.67
	16	0.40	0.00	2.15	0.00
	32	99.32	99.96	0.24	0.00
	$\geq$48	99.10	99.75	0.34	0.00
	$\geq$144	98.94	0.00	97.32	0.00
$\geq$35	0	0.18	0.00	99.37	0.00
	16	0.37	11.11	91.87	0.00
	32	0.39	78.95	89.89	0.00
	48	1.54	68.00	95.58	0.00
	64	3.22	98.73	89.83	0.00
	128	60.35	99.96	39.24	0.00
	$\geq$144	95.33	0.00	99.64	0.00

mission of the *low energy* frame from the LE node before the *high energy* frame from the HE node, and to configure the exact delay Δt as a multiple of the micro-slot duration.

For instance, we set up a $\Delta t = 32$ µs by configuring LE node to transmit at slot 15, HE node at slot 17. By moving LE node away from the *ath* node while the HE node is always close, we are able to control the relative power difference ΔP received by the *ath* node between frames coming from the LE and HE nodes. With the configured timings, we are able to replicate the reception experiment at the *ath* node approximately 976 times per second, thus collecting meaningful statistics in seconds.

We obtained the results shown in Table 1. When the energy gap is small ($\leq$5 dB), the MIM effect never enters into play as we can see from the first part of Table 1. If the two frames are transmitted at the same time, then the QCA card receives the majority of the HE frames (92%) despite some of them are broken (17%); almost no LE frames are received. By increasing the delay to 16 µs, the QCA card stops working: the short delay means that the HE frame collide with the LE one at the PLCP level. The energy gap does not allow the QCA correlator to restart decoding a new PLCP and, in fact, only a few frames are sporadically received. Further increasing the delay allows the QCA card to correctly receive the PLCP preamble of the LE frame, but then the PDU decoding is affected by errors (e.g., delay set to 48 µs) because of collision. Finally, if the delay is high enough so that both frames fit into a schedule, the QCA card receives everything correctly ($\geq$144 µs).

When the energy gap exceeds a threshold (e.g., more than 35 dB), then the behaviour of the QCA card changes radically as we can see from the second part of Table 1: first, with no delay, all high energy frames are received (expected given that they overkill the others); second, when both frame types fit in the schedule, all of them are received, which confirms that the link between LE node and the QCA is still very good. But, unlike the previous case, HE frames are received regardless of the delay, which means that the correlator restarts decoding the PLCP of the second frame because of the higher energy, enough for distinguishing it from the first frame that simply turns into a negligible noise.

Thus, our experiments confirm that the MIM effect actually affects modern wireless cards, and therefore it should be taken into account in any micro-sleep strategy. Let us consider, for instance, a common infrastructure-based scenario in which certain STA receives low-power frames from a distant network in the same channel. If the AP does not see them, we are facing the hidden node problem. It is clear that none of these frames will be addressed to our STA, but, if it goes to sleep during these transmissions, it may lose potential high-power frames from its BSSID. Therefore, if we perform micro-sleeps under hidden node conditions, in some cases we may lose frames that we would receive otherwise thanks to the capture effect. The same situation may happen within the local BSSID (the low-power frames belong to the same network), but this is far more rare, as such a hidden node will become disconnected sooner or later.

In order to circumvent these issues, a STA should only exploit micro-sleep opportunities arising from its own network. To discard packets originating from other networks, the algorithm looks at the BSSID in the receiver address within frames addressed to an AP. If the frame was sent by an AP, it only needs to read 6 additional bytes (in the worst case), which are included in the transmitter address. Even so, these additional bytes do not necessarily involve consuming more time, depending on the modulation. For instance, for OFDM 11ag rates, this leads to a time increase of 8 µs at 6 and 9 Mbps, 4 µs at 12, 18 and 36 Mbps, and no time increase at 24, 48 and 54 Mbps.

4.2.2. Impact of errors in the MAC header

Taking decisions without checking the FCS (placed at the end of the frame) for errors or adding any protection mechanism may lead to performance degradation due to frame loss. This problem was firstly identified by Balaji et al. [11], Prasad et al. [12] which, based on purely qualitative criteria, reached opposite conclusions. The first work advocates for the need for a new CRC to protect the header bits while the latter dismisses this need. This section is devoted to analyse quantitatively the impact of errors.

At a first stage, we need to identify, field by field, which cases are capable of harming the performance of our algorithm due to frame loss. The duration/ID field (2 bytes) and the MAC addresses (6 bytes each) are an integral part of our algorithm. According to its encoding, the duration/ID field will be interpreted as an actual duration if and only if the bit 15 is equal to 0. Given that the bit 15 is the most significant one, this condition is equivalent to the value being smaller than 32,768. Therefore, we can distinguish the following cases in terms of the possible errors:

- *An error changes the bit 15 from 0 to 1*. The field will not be interpreted as a duration and hence we will not go to sleep. We will be missing an opportunity to save energy, but there will be no frame loss and, therefore, the network performance will not be affected.
- *An error changes the bit 15 from 1 to 0*. The field will be wrongly interpreted as a duration. The resulting *sleep* will be up to 33 ms longer than required, with the potential frame loss associated.
- *With the bit 15 equal to 0, an error affects the previous bits*. The resulting sleep will be shorter or longer that the real one. In the first case, we will be missing an opportunity to save energy; in the second case, there is again a potential frame loss.

Regarding the receiver address field, there exist the following potential issues:

- *A multicast address changes but remains multicast*. The frame will be received and discarded, i.e., the behaviour will be the same as with no error. Hence, it does not affect.
- *A unicast address changes to multicast*. The frame will be received and discarded after detecting the error. If the unicast frame was addressed to this host, it does not affect. If it was addressed to another host, we will be missing an opportunity to save energy.
- *A multicast address changes to unicast*. If the unicast frame is addressed to this host, it does not affect. If it is addressed to another host, we will save energy with a frame which would be otherwise received and discarded.
- *Another host's unicast address changes to your own*. This case is very unlikely. The frame will be received and discarded, so we will be missing an opportunity to save energy.
- *Your own unicast address changes to another's*. We will save energy with a frame otherwise received and discarded.

As for the transmission address field, this is checked as an additional protection against the undesirable effects of the already discussed intra-frame capture effect. If the local BSSID in a packet changes to another BSSID, we will be missing an opportunity to save energy. It is extremely unlikely that an error in this field could lead to frame loss: a frame from a foreign node (belonging to another BSSID and hidden to our AP) should contain an error that matches the local BSSID in the precise moment in which our AP tries to send us a frame (note that this frame might be received because of the MIM effect explained previously).

Henceforth, we draw the following conclusions from the above analysis:

- Errors at the MAC addresses *do not produce frame loss*, because under no circumstances they imply frame loss. The only impact is that there will be several new opportunities to save energy and several others will be wasted.
- Errors at the duration/ID field, however, *may produce frame loss* due to frame loss in periods of time up to 33 ms. Also several energy-saving opportunities may be missed without yielding any frame loss.
- An error burst affecting both the duration/ID field and the receiver address may potentially change the latter in a way that the frame would be received (multicast bit set to 1) and discarded, and thus preventing the frame loss.

From the above, we have that the only case that may yield performance degradation in terms of frame loss is when we have errors in the duration/ID field. In the following, we are going to analytically study and quantify the probability of frame loss in this case. For our analysis, we first consider statistically independent single-bit errors. Each bit is considered the outcome of a Bernoulli trial with a success probability equal to the bit error probability p_b. Thus, the number of bit errors, X, in certain field is given by a Binomial distribution $X \sim \mathrm{B}(N, p_b)$, where N is the length of that field.

With these assumptions, we can compute the probability of having more than one erroneous bit, $\Pr(X \geq 2)$, which is three-four orders of magnitude smaller than p_b with realistic p_b values. Therefore, we assume that we never have more than one bit error in the frame header, so the probability of receiving an erroneous duration value with a single-bit error, $p_{e,\,b}$, is the following:

$$p_{e,b} \approx 1 - (1 - p_b)^{15} \tag{6}$$

However, not all the errors imply a duration value greater than the original one, but only those which convert a zero into a one. Let us call $\mathrm{Hw}(i)$ the Hamming weight, i.e., the number of ones in the binary representation of the integer i. The probability of an erroneous duration value greater than the original, $p_{eg,\,b}$, is the following:

$$p_{eg,b}(i) = p_{e,b} \cdot \frac{15 - \mathrm{Hw}(i)}{15} \tag{7}$$

which represents a fraction of the probability $p_{e,\,b}$ and depends on the original duration i (before the error).

In order to understand the implications of the above analysis into real networks, we have analysed the data set SIGCOMM'08 [23] and gathered which duration values are the most common. In the light of the results depicted in Table 2, it seems reasonable to approximate $p_{eg,\,b}/p_{e,\,b} \approx 1$, because it is very likely that the resulting duration will be greater than the original.

Finally, we can approximate p_b by the BER and, based on the above data and considerations, the frame loss probability, p_{loss}, due to an excessive sleep interval using a single-bit error model is the following:

$$p_{\mathrm{loss}} = p_{eg,b} \approx p_{e,b} \approx 1 - (1 - \mathrm{BER})^{15} \tag{8}$$

Table 2
Most frequent duration values.

Duration	%	$p_{eg,\,b}/p_b$	Cause
44	62.17	0.88	SIFS + ACK at 24 Mbps
0	25.23	1.00	Broadcast, multicast... packets
60	6.54	0.73	SIFS + ACK at 6 Mbps
48	5.82	0.87	SIFS + ACK at 12 Mbps

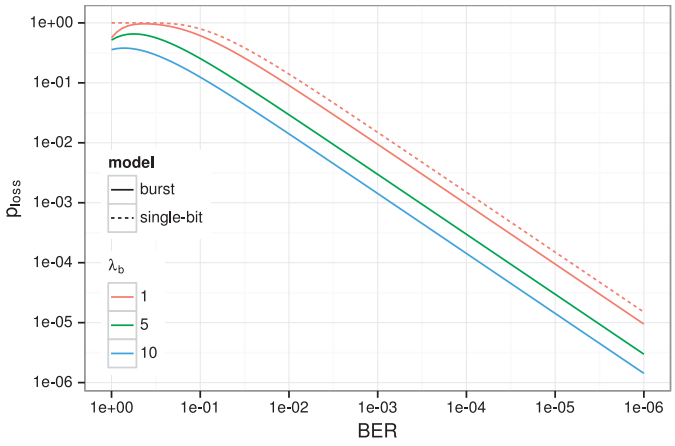

Fig. 5. Frame loss probability given a BER level.

The above analysis assumes that errors occur independently. However, it is well known that in reality errors typically occur in bursts. In order to understand the impact of error bursts in our scheme, we analyse a scenario with independent error bursts of length X bits, where X is a random variable. To this end, we use the Neyman-A contagious model [24], which has been successfully applied in telecommunications to describe burst error distributions [25–27]. This model assumes that both the bursts and the burst length are Poisson-distributed. Although assuming independency between errors in the same burst may not be accurate, it has been shown that the Neyman-A model performs well for short intervals [28], which is our case.

The probability of having k errors in an interval of N bits, given the Neyman-A model, is the following:

$$p_N(k) = \frac{\lambda_b^k}{k!} e^{-\lambda_B} \sum_{i=0}^{\infty} \frac{i^k}{i!} \lambda_B^i e^{-i\lambda_b} \tag{9}$$

where

λ_b is the average number of bits in a burst.
$\lambda_B = Np_b/\lambda_b$ is the average number of bursts.

This formula can be transformed into a recursive one with finite sums [24]:

$$p_N(k) = \frac{\lambda_B \lambda_b e^{-\lambda_b}}{k} \sum_{j=0}^{k-1} \frac{\lambda_b^j}{j!} p_N(k-1-j)$$

$$p_N(0) = e^{-\lambda_B\left(1-e^{-\lambda_b}\right)} \tag{10}$$

Following the same reasoning as for the single-bit case, we can assume one burst at a time which will convert the duration value into a higher one. Then, the frame loss probability is the following:

$$p_{\mathrm{loss}} = \sum_{k=1}^{15} p_{15}(k) \tag{11}$$

with parameters λ_b and $p_b \approx \mathrm{BER}$.

Fig. 5 evaluates both error models as a function of BER. As expected, the single-bit error model is an upper bound for the error burst model and represents a worst-case scenario. At most, the

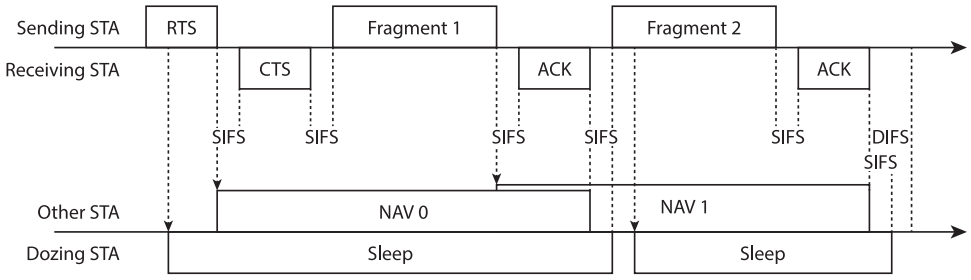

Fig. 6. RTS/CTS-based fragmented transmission example and μNap's behaviour.

frame loss probability is one order of magnitude higher than BER. Therefore, we conclude that the frame loss is negligible for reasonable BERs and, consequently, the limited benefit of an additional CRC does not compensate the issues.

4.3. Algorithm design

In the following, we present μNap, which builds upon the insights provided in previous sections and tries to save energy during the channel transmissions in which the STA is not involved. However, not all transmissions addressed to other stations are eligible for dozing, as the practical issues derived from the capture effect may incur in performance degradation. Therefore, the algorithm must check both the receiver as well as the transmitter address in the MAC header in order to determine whether the incoming frame is addressed to another station *and* it comes from within the same network.

If these conditions are met, a basic micro-sleep will last the duration of the rest of the incoming frame plus an inter-frame space (SIFS). Unfortunately, the long times required to bring an interface back and forth from sleep, as discovered in Section 3, shows that this basic micro-sleep may not be long enough to be exploitable. Thus, the algorithm should take advantage of the NAV field whenever possible. Our previous analysis shows that this duration information stored in the NAV is not exploitable in every circumstance: the interface can leverage this additional time during CPs and it must avoid any NAV set by a CTS packet.

Finally, after a micro-sleep, two possible situations arise:

- The card wakes up at the end of a frame exchange. For instance, after a data + ACK exchange. In this case, all STAs should wait for a DIFS interval before contending again.
- The card wakes up in the middle of a frame exchange. For instance, see Fig. 6, where an RTS/CTS-based fragmented transmission is depicted.

In the latter example, an RTS sets the NAV to the end of a fragment, and our algorithm triggers the sleep. This first fragment sets the NAV to the end of the second fragment, but it is not seen by the dozing STA. When the latter wakes up, it sees a SIFS period of silence and then the second fragment, which sets its NAV again and may trigger another sleep. This implies that the STA can doze for an additional SIFS, as Fig. 6 shows, and wait in idle state until a DIFS is completed before trying to contend again.

Based on the above, Algorithm 1 describes the main loop of a wireless card's microcontroller that would implement our mechanism. When the first 16 bytes of the incoming frame are received, all the information needed to take the decision is available: the duration value (Δt_{NAV}), the receiver address (R_A) and the transmitter address (T_A). The ability to stop a frame's reception at any point has been demonstrated to be feasible [29]. Note that MAC addresses can be efficiently compared in a streamed way, so that the first differing byte (if the first byte of the R_A has the multicast bit set to zero, i.e., R_A is unicast) triggers our sleep procedure

Algorithm 1 μNap implementation: main loop modification for energy saving during packet overhearing.

```
1:  ...                                                    ▷ Initialisation
2:  global C ← true                                        ▷ Contention flag
3:  loop                                                   ▷ Main loop
4:      ...
5:      while bytes remaining do                           ▷ Receiving loop
6:          READ
7:          if R_A = BSSID OR (T_A = BSSID AND
                R_A is other unicast MAC) then
8:              SET_SLEEP(Δt_DATA, Δt_NAV)
9:          end if
10:     end while
11:     CHECK_FCS                                          ▷ Frame received
12:     if is Beacon AND Δt_NAV > 0 then                   ▷ CFP starts
13:         C ← false
14:     else if is CF_End then                             ▷ CFP ends
15:         C ← true
16:     end if
17:     ...
18: end loop
19: procedure SET_SLEEP(Δt_DATA, Δt_NAV)
20:     Δt_sleep ← Δt_DATA + Δt_SIFS
21:     if C AND is not CTS AND Δt_NAV ≤ 32767 then
22:         Δt_sleep ← Δt_sleep + Δt_NAV
23:     end if
24:     if Δt_sleep ≥ Δt_sleep,min then
25:         SLEEP(Δt_sleep)
26:         WAIT(Δt_DIFS − Δt_SIFS)
27:         go to Main loop
28:     end if
29:     go to Receiving loop
30: end procedure
```

(SET_SLEEP in Algorithm 1). In addition, the main loop should keep up to date a global variable (C) indicating whether the contention is currently allowed (CP) or not (CFP). This is straightforward, as every CFP starts and finishes with a beacon frame.

The SET_SLEEP procedure takes as input the remaining time until the end of the incoming frame (Δt_{DATA}) and the duration value (Δt_{NAV}). The latter is used only if it is a valid duration value and a CP is active. Then, the card may doze during Δt_{sleep} (if this period is greater than $\Delta t_{\text{sleep, min}}$), wait for a DIFS to complete and return to the main loop.

Finally, it is worth noting that this algorithm is deterministic, as it is based on a set of conditions to trigger the sleep procedure. It works locally with the information already available in the protocol headers, without incurring in any additional control overhead

and without impacting the normal operation of 802.11. Specifically, our analytical study of the impact of errors in the first 16 bytes of the MAC header shows that the probability of performance degradation is comparable to the BER under normal channel conditions. Therefore, the overall performance in terms of throughput and delay is completely equivalent to normal 802.11.

5. Performance evaluation

This section is devoted to evaluate the performance of μNap. First, Section 5.1, through trace-driven simulation, shows that μNap significantly reduces the overhearing time and the energy consumption of a real network. Secondly, Section 5.2 analyses the impact of the timing constraints imposed by the hardware, which are specially bad in the case of the AR9280, and discusses the applicability of μNap in terms of those parameters and the evolution trends in the 802.11 standard.

5.1. Evaluation with real traces

In the following, we conduct an evaluation to assess how much energy might be saved in a real network if all STAs implement μNap using the AR9280, the wireless card characterised in Section 3. The reasons for this are twofold. On the one hand, the timing properties of this interface are particularly bad if we think of typical frame durations in 802.11, which means that many micro-sleep opportunities will be lost due to hardware constraints. On the other hand, it does not support newer standards that could potentially lead to longer micro-sleep opportunities through mechanisms such as frame aggregation. Therefore, an evaluation based on an 11a/g network and the AR9280 chip represents a worst case scenario for our algorithm.

For this purpose, we used 802.11a wireless traces with about 44 million packets, divided in 43 files, from the data set SIGCOMM'08 [23]. The methodology followed to parse each trace file is as follows. Firstly, we discover all the STAs and APs present. Each STA is mapped into its BSSID and a bit array is developed in order to hold the status at each point in time (online or offline). It is hard to say when a certain STA is offline from a capture, because they almost always disappear without sending a disassociation frame. Thus, we use the default rule in `hostapd`, the daemon that implements the AP functionality in Linux: a STA is considered online if it transmitted a frame within the last 5 min.

Secondly, we measure the amount of time that each STA spends (without our algorithm) in the following states: transmission, reception, overhearing and idle. We consider that online STAs are always awake; i.e., even if a STA announces that it is going into PS mode, we ignore this announcement. We measure also the amount of time that each STA would spend (with our algorithm) in transmission, reception, overhearing, sleep and idle. Transmission and reception times match the previous case, as expected. As part of idle time, we account separately the wasted time in each micro-sleep as a consequence of hardware limitations (the fixed toll Δt_{waste}). After this processing, there are a lot of duplicate unique identifiers (MAC addresses), i.e., STAs appearing in more than one trace file. Those entries are summarised by aggregating the time within each state.

At this point, let us define the activity time as the sum of transmission, reception, overhearing, sleep and wasted time. We do not account for idle time since our goal is to understand how much power we can save in the periods of activity, which are the only ones that consume power in wireless transmissions (the scope of this paper). Using the definition above, we found that the majority of STAs reveals very little activity (they are connected for a few seconds and disappear). Therefore, we took the upper decile in terms of activity, thus obtaining the 42 more active STAs.

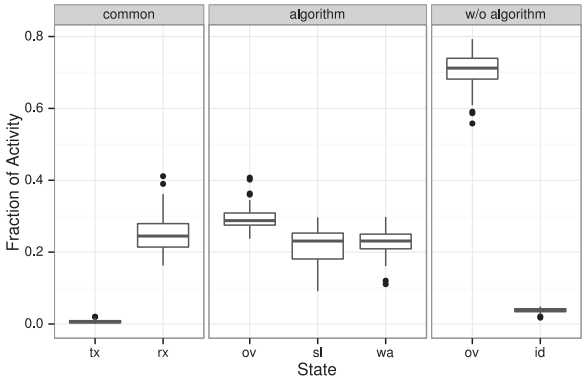

Fig. 7. Normalised activity aggregation of all STAs.

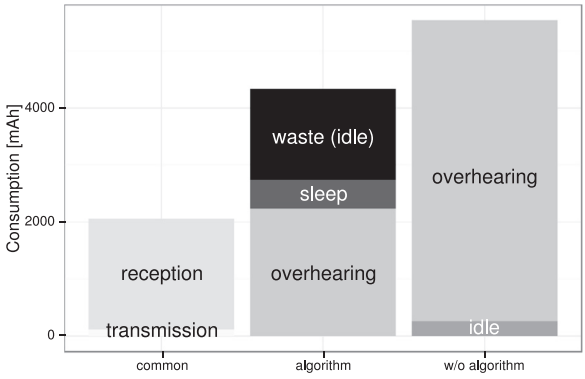

Fig. 8. Energy consumption aggregation of all STAs.

The activity aggregation of all STAs is normalised and represented in Fig. 7. Transmission (tx) and reception (rx) times are labelled as common, because STAs spend the same time transmitting and receiving both with and without our algorithm. It is clear that our mechanism effectively reduces the total overhearing (ov) time from a median of 70% to a 30% approximately (a 57% reduction). The card spends consistently less time in overhearing because this overhearing time difference, along with some idle (id) time from inter-frame spaces, turns into micro-sleeps, that is, sleep (sl) and wasted (wa) time.

This activity aggregation enables us to calculate the total energy consumption using the power values from the thorough characterisation presented in A.1. Fig. 8 depicts the energy consumption in units of mAh (assuming a typical 3.7-V battery). The energy savings overcome 1200 mAh even with the timing limitations of the AR9280 card, which (1) prevents the card from going to sleep when the overhearing time is not sufficiently long, and (2) wastes a long fixed time in idle during each successful micro-sleep. This reduction amounts to a 21.4% of the energy spent in overhearing and a 15.8% of the total energy during the activity time, when the transmission and reception contributions are also considered.

Fig. 9 provides a breakdown of the data by STA. The lower graph shows the activity breakdown per STA for our algorithm (transmission bars, in white, are very small). Overhearing time is reduced to a more or less constant fraction for all STAs (i.e., with the algorithm, the overhearing bars represent more or less a 30% of the total activity for all STAs), while less participative STAs (left part of the graph) spend more time sleeping. The upper graph shows the energy consumption per STA with our algorithm along with the energy-saving in dark gray, which is in the order of tens of mAh per STA.

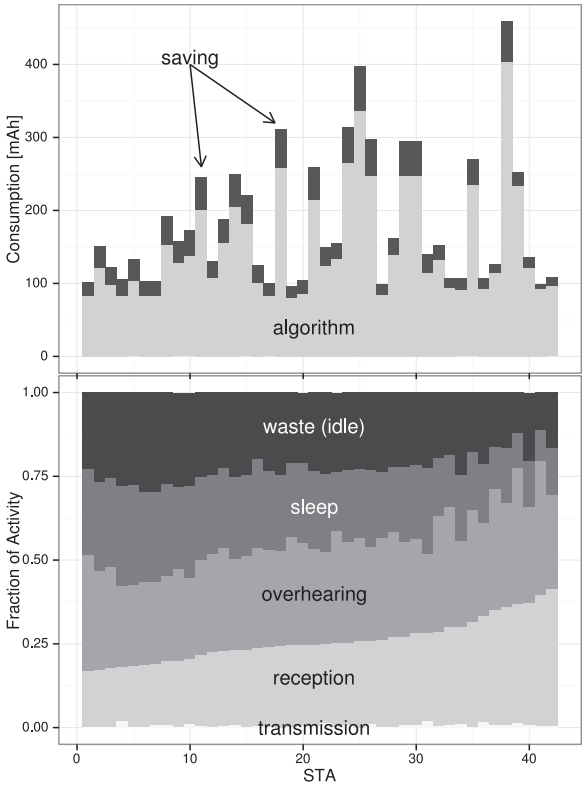

Fig. 9. Energy consumption (up) and normalised activity (bottom) for each STA.

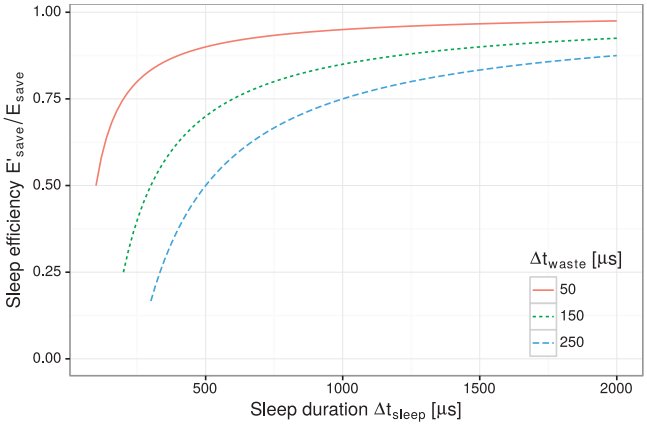

Fig. 10. Sleep efficiency behaviour as Δt_{waste} decreases.

5.2. Impact of timing constraints

The performance gains of μNap depend on the behaviour of the circuitry. Its capabilities, in terms of timing, determine the maximum savings that can be achieved. Particularly, each micro-sleep has an efficiency (in comparison to an ideal scheme in which the card stays in sleep state over the entire duration of the micro-sleep) given by

$$\frac{E'_{\text{save}}}{E_{\text{save}}} = \frac{E_{\text{save}} - E_{\text{waste}}}{E_{\text{save}}} \approx 1 - \frac{\Delta t_{\text{waste}}}{\Delta t_{\text{sleep}}} \tag{12}$$

which results from the combination of Eqs. (1), (2) and (5).

Fig. 10 represents this sleep efficiency for the AR9280 card ($\Delta t_{\text{waste}} = 250$) along with other values. It is clear that an improvement of Δt_{waste} is fundamental to boost performance in short sleeps.

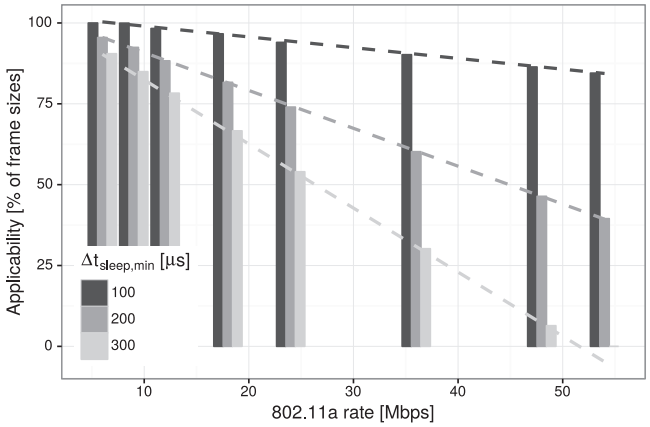

Fig. 11. Algorithm applicability for common transmissions (≤1500 bytes + ACK) in 802.11a DCF mode.

Similarly, the constraint $\Delta t_{\text{sleep, min}}$ limits the applicability of μNap, especially in those cases where the NAV cannot be used to extend the micro-sleep. For instance, let us consider the more common case in 11a/b/g networks: the transmission of a frame (up to 1500 bytes long) plus the corresponding ACK. Then,

$$\Delta t_{\text{sleep,min}} \leq \Delta t_{\text{DATA}} + \Delta t_{\text{SIFS}} + \Delta t_{\text{ACK}} + \Delta t_{\text{SIFS}} \tag{13}$$

and expanding the right side of the inequality,

$$\Delta t_{\text{sleep,min}} \leq \frac{8(14 + l_{\text{min}} + 4)}{\lambda_{\text{DATA}}} + \Delta t_{\text{SIFS}} + \Delta t_{\text{PLCP}} + \frac{8(14+2)}{\lambda_{\text{ACK}}} + \Delta t_{\text{SIFS}} \tag{14}$$

Here, we can find l_{min}, which is the minimum amount of data (in bytes, and apart from the MAC header and the FCS) that a frame must contain in order to last $\Delta t_{\text{sleep, min}}$. Based on this l_{min}, Fig. 11 defines the applicability in 802.11a DCF in terms of frame sizes ($\leq$ 1500 bytes) that last $\Delta t_{\text{sleep, min}}$ at least. Again, an improvement in Δt_{waste} would boost not only the energy saved per sleep, but also the general applicability defined in this way.

The applicability of μNap may also be affected by the evolution of the standard. Particularly, 802.11n introduced, and 802.11ac followed, a series of changes enabling high and very high throughput respectively, up to Gigabit in the latter case. This improvement is largely based on MIMO and channel binding: multiple spatial and frequency streams. Nevertheless, a single 20-MHz spatial stream is more or less equivalent to 11ag. Some enhancements (shorter guard interval and coding enhancements) may boost the throughput of a single stream from 54 to 72 Mbps under optimum conditions. Yet it is also the case that the PLCP is much longer to accommodate the complexity of the new modulation coding schemes (MCSs). This overhead not only extends each transmission, but also encourages the use of frame aggregation. Thus, the increasing bandwidth, in current amendments or future ones, does not necessarily imply a shorter airtime in practice, and our algorithm is still valid.

Reducing PHY's timing requirements is essential to boost energy savings, but its feasibility should be further investigated. Nonetheless, there are some clues that suggest that there is plenty of room for improvement. In the first place, Δt_{off} and Δt_{on} should depend on the internal firmware implementation (i.e., the complexity of the saved/restored state). Secondly, Fig. 3(a) indicates that a transmission is far more aggressive, in terms of a sudden power rise, than a return from sleep. From this standpoint, $\Delta t_{\text{ready}} = 200$ µs would be a pessimistic estimate of the time required by the circuitry to stabilise. Last, but not least, the 802.3

standard goes beyond 802.11 and, albeit to a limited extent, it defines some timing parameters of the PHYs (e.g., Δt_{w_phy} would be equivalent to our $\Delta t_{on} + \Delta t_{ready}$). These timing parameters are in the range of tens of μs in the worst case (see Table 78-4 of the IEEE Std 802.3-2008 [30]).

Due to these reasons, WiFi card manufacturers should push for a better power consumption behaviour, which is necessary to boost performance with the power-saving mechanism presented in this paper. Furthermore, it is necessary for the standardisation committees and the manufacturers to collaborate to agree power consumption behaviour guidelines for the hardware (similarly to what has been done with 802.3). Indeed, strict timing parameters would allow researchers and developers to design more advanced power-saving schemes.

6. Conclusions

Based on a thorough characterisation of the timing constraints and energy consumption of 802.11 interfaces, we have exhaustively analysed the micro-sleep opportunities that are available in current WLANs. We have unveiled the practical challenges of these opportunities, previously unnoticed in the literature, and, building on this knowledge, we have proposed μNap an energy-saving scheme that is orthogonal to the existing standard PS mechanisms. Unlike previous attempts, our scheme takes into account the non-zero time and energy required to move back and forth between the active and sleep states, and decides when to put the interface to sleep in order to make the most of these opportunities while avoiding frame losses.

We have demonstrated the feasibility of our approach using a robust methodology and high-precision instrumentation, showing that, despite the limitations of COTS hardware, the use of our scheme would result in a 57% reduction in the time spent in overhearing, thus leading to an energy saving of 15.8% of the activity time according to our trace-based simulation. Finally, based on these results, we have made the case for the strict specification of energy-related parameters of 802.11 hardware, which would enable the design of platform-agnostic energy-saving strategies.

Appendix A. Energy consumption characterisation

A1. State consumption parametrisation

In order to gain insight into the energy savings of μNap, we performed a complete state parametrisation (power consumption in transmission, reception, overhearing, idle and sleep) of the AR9280 card (the active state in the traces used for the evaluation, Section 5.1) using the same scenario as in Section 3 (see Fig. 2). As in Section 3, all measurements (except for the sleep state) were taken with the wireless card associated to the AP in 11a mode to avoid any interfering traffic, and it was placed very close to the node to obtain the best possible signal quality. The reception of beacons is accounted in the baseline consumption (idle).

The card under test performed transmissions/receptions to/from the AP at a constant rate and with fixed packet length. In order to avoid artifacts from the reception/transmission of ACKs, UDP was used and the NoACK policy was enabled. Packet overhearing was tested by generating traffic of the same characteristics from a secondary STA placed in the same close range (∼cm). Under these conditions, several values of airtime percentage were swept. For each experiment, current and voltage signals were sampled at 100 kHz and the mean power consumption was measured with a basic precision of 1 mW over intervals of 3 s.

Regarding the sleep state, the driver `ath9k` internally defines three states of operation: *awake, network sleep* and *full sleep*. A closer analysis reveals that the card is awake, or in active state, when it is operational (i.e., transmitting, receiving or in idle state, whether as part of an SSID or in monitor mode), and it is in full sleep state when it is not operational at all (i.e., interface down or up but not connected to any SSID). The network sleep state is used by the 802.11 PS mechanism, but essentially works in the same way as full sleep, that is, it turns off the main reference clock and switches to a secondary 32 kHz one. Therefore, we saw that full sleep and network sleep are the same state in terms of energy: they consume exactly the same power. The only difference is that network sleep sets up a tasklet to wake the interface periodically (to receive TIMs), as required by the PS mode.

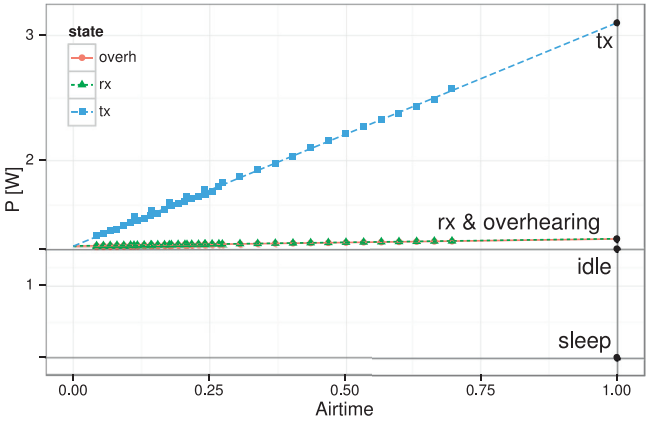

Fig. A1. Atheros AR9280 power consumption in 11a mode.

Table A1
Atheros AR9280 power consumption.

State	Mode	Channel	MHz	Power [W]
Transmission	11a	44	20	3.10(2)
Reception	11a	44	20	1.373(1)
Overhearing	11a	44	20	1.371(1)
Idle	11a	44	20	1.292(2)
Sleep	–	–	–	0.424(2)
Idle	11n	11	20	1.137(4)
Idle	11n	11	40	1.360(4)

Fig. A.12 shows our results for transmission, reception and overhearing. Idle and sleep consumptions were measured independently, are depicted with gray horizontal lines for reference. As expected, power consumptions in transmission/reception/overhearing state are proportional to airtime, thus the power consumption of such operations can be easily estimated by extrapolating the regression line to the 100% of airtime (gray vertical line).

These mean values are shown in Table A.3. First of all, reception and overhearing consumptions are the same within the error, and they are close to idle consumption. Transmission power is more than two times larger than reception. Finally, the sleep state saves almost the 70% of the energy compared to idle/reception.

A2. Downclocking consumption characterisation

As the AR9280's documentation states, its reference clock runs at 44 MHz for 20 MHz channels and at 88 MHz for 40 MHz channels in the 2.4 GHz band, and at 40 MHz for 20 MHz channels and at 80 MHz for 40 MHz channels in the 5 GHz band. Thus, as Table A.3 shows, we measured two more results to gain additional insight into the behaviour of the main reference clock, which is known to be linear [6].

Using an 11n-capable AP, we measured the idle power in the 2.4 GHz band with two channel widths, 20 and 40 MHz. Note that the idle power in 11a mode (5 GHz band), with a 40 MHz clock, is higher than the idle power with a 44 MHz clock. This is

because both bands are not directly comparable, as the 5 GHz one requires more amplification (the effect of the RF amplifier is out of the scope of this paper).

With these two points, we can assume a higher error (of about 10 mW) and try to estimate a maximum and a minimum slope for the power consumed by the main clock as a function of the frequency f. The resulting averaged regression formula is the following:

$$P(f) = 0.91(3) + 0.0051(5)f \quad \text{(A.1)}$$

This result, although coarse, enables us to estimate how a downclocking approach should perform in COTS devices. It shows that the main consumption of the clock goes to the baseline power (the power needed to simply turn it on), and that the increment per MHz is low: 5.1(5) mW/MHz. As a consequence, power-saving mechanisms based on idle downclocking, such as [6], will not save too much energy compared to the sleep state of COTS devices. For instance, the x16 downclock of [6] applied to this Atheros card throws an idle power consumption of 1.10(2) W in 11a mode, i.e., about a 15% of saving according to Table A.3, which is low compared to the 70% of its sleep state. This questions the effectiveness of complex schemes based on downclocking compared to simpler ones based on the already existing sleep state.

References

[1] L. Feeney, M. Nilsson, Investigating the energy consumption of a wireless network interface in an ad hoc networking environment, in: Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No.01CH37213), 3, IEEE, 2001, pp. 1548–1557, doi:10.1109/INFCOM.2001.916651.

[2] X. Perez-Costa, D. Camps-Mur, IEEE 802.11E QoS and power saving features overview and analysis of combined performance [Accepted from Open Call], IEEE Wireless Commun. 17 (4) (2010) 88–96, doi:10.1109/MWC.2010.5547926.

[3] P. Basu, J. Redi, Effect of overhearing transmissions on energy efficiency in dense sensor networks, in: Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks, in: IPSN '04, ACM, New York, NY, USA, 2004, pp. 196–204, doi:10.1145/984622.984652.

[4] J. Liu, L. Zhong, Micro power management of active 802.11 interfaces, in: Proceeding of the 6th International Conference on Mobile Systems, Applications, and Services - MobiSys '08, ACM Press, New York, New York, USA, 2008, p. 146, doi:10.1145/1378600.1378617.

[5] K.-Y. Jang, S. Hao, A. Sheth, R. Govindan, Snooze: energy management in 802.11n WLANs, in: Proceedings of the Seventh Conference on Emerging Networking Experiments and Technologies on - CoNEXT '11, ACM Press, New York, New York, USA, 2011, pp. 1–12, doi:10.1145/2079296.2079308.

[6] X. Zhang, K.G. Shin, E-MiLi: energy-minimizing idle listening in wireless networks, IEEE Trans. Mob. Comput. 11 (9) (2012) 1441–1454, doi:10.1109/TMC.2012.112.

[7] A. Kamerman, L. Monteban, WaveLAN-II: a high-performance wireless LAN for the unlicensed band, Bell Labs Tech. J. 2 (3) (1997) 118–133, doi:10.1002/bltj.2069.

[8] P.J. Havinga, G.J. Smit, Energy-efficient TDMA medium access control protocol scheduling, in: Asian International Mobile Computing Conference, AMOC, 2000, pp. 1–10.

[9] E.-S. Jung, N.H. Vaidya, An energy efficient MAC protocol for wireless lans, in: INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE, 3, 2002, pp. 1756–1764, doi:10.1109/INFCOM.2002.1019429. vol. 3.

[10] V. Baiamonte, C.-F. Chiasserini, Saving energy during channel contention in 802.11 WLANs, Mobile Netw. Appl. 11 (2) (2006) 287–296, doi:10.1007/s11036-006-4480-x.

[11] B. Balaji, B.R. Tamma, B.S. Manoj, A novel power saving strategy for greening IEEE 802.11 based wireless networks, in: 2010 IEEE Global Telecommunications Conference GLOBECOM 2010, IEEE, 2010, pp. 1–5, doi:10.1109/GLOCOM.2010.5684071.

[12] R. Prasad, A. Kumar, R. Bhatia, B.R. Tamma, Ubersleep: an innovative mechanism to save energy in IEEE 802.11 based WLANs, in: 2014 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT), IEEE, 2014, pp. 1–6, doi:10.1109/CONECCT.2014.6740349.

[13] R. Palacios, F. Granelli, D. Kliazovich, L. Alonso, J. Alonso-Zarate, An energy efficient distributed coordination function using bidirectional transmissions and sleep periods for IEEE 802.11 WLANs, in: 2013 IEEE Global Communications Conference (GLOBECOM), 2013, pp. 1619–1625, doi:10.1109/GLOCOM.2013.6831305.

[14] R. Palacios, F. Granelli, D. Gajic, C. Liß, D. Kliazovich, An energy-efficient point coordination function using bidirectional transmissions of fixed duration for infrastructure IEEE 802.11 WLANs, in: 2013 IEEE International Conference on Communications (ICC), 2013, pp. 2443–2448, doi:10.1109/ICC.2013.6654898.

[15] R. Palacios-Trujillo, J. Alonso-Zarate, F. Granelli, F.H.P. Fitzek, N.L.S. da Fonseca, Network coding and duty cycling in ieee 802.11 wireless networks with bidirectional transmissions and sleeping periods, in: 2015 IEEE Global Communications Conference (GLOBECOM), 2015, pp. 1–7, doi:10.1109/GLOCOM.2015.7417689.

[16] R. Palacios, G.M. Mekonnen, J. Alonso-Zarate, D. Kliazovich, F. Granelli, Analysis of an energy-efficient MAC protocol based on polling for IEEE 802.11 WLANs, in: 2015 IEEE International Conference on Communications (ICC), 2015, pp. 5941–5947, doi:10.1109/ICC.2015.7249269.

[17] Using a Unity Gain Buffer (Voltage Follower) with a DAQ Device, 2014, (White paper, National Instruments).

[18] IEEE Standard for information technology–telecommunications and information exchange between systems local and metropolitan area networks–specific requirements part 11: wireless LAN medium access control (MAC) and physical layer (PHY) specifications, IEEE Std 802.11–2012 (Revision of IEEE Std 802.11–2007) (2012) 1–2793. doi:10.1109/IEEESTD.2012.6178212.

[19] J. Lee, W. Kim, S.-J. Lee, D. Jo, J. Ryu, T. Kwon, Y. Choi, An experimental study on the capture effect in 802.11a networks, in: Proceedings of theSecond ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization - WinTECH '07, ACM Press, New York, New York, USA, 2007, p. 19, doi:10.1145/1287767.1287772.

[20] N. Santhapuri, R.R. Choudhury, J. Manweiler, S. Nelakuduti, S. Sen, K. Munagala, Message in message mim: a case for reordering transmissions in wireless networks, In HotNets, 2008.

[21] W. Wang, W.K. Leong, B. Leong, Potential pitfalls of the message in message mechanism in modern 802.11 networks, in: Proceedings of the 9th ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization, in: WiNTECH '14, ACM, New York, NY, USA, 2014, pp. 41–48, doi:10.1145/2643230.2643231.

[22] OpenFWWF - Open FirmWare for WiFi networks, 2015, (Website: http://www.ing.unibs.it/~openfwwf/).

[23] CRAWDAD data set umd/sigcomm2008 (v. 2009-03-02), 2009, (Downloaded from http://crawdad.org/umd/sigcomm2008/).

[24] J. Neyman, On a new class of 'contagious' distributions, applicable in entomology and bacteriology, Ann. Math. Stat. 10 (1) (1939) 35–57.

[25] ITU-R, Allowable error performance for a satellite hypothetical reference digital path in the fixed-satellite service operating below 15 GHz when forming part of an international connection in an integrated services digital network, S-Series: Fixed-satellite service, International Telecommunication Union, 2005. ITU-R Recommendation S.614.

[26] D. Becam, P. Brigant, R. Cohen, J. Szpirglas, Validité du modèle de neyman pour les processus derreurs sur des liaisons numériques à 2 et 140 mbit/s, in: Annales des Télécommunications, 40, Springer, 1985, pp. 17–25.

[27] D.R. Irvin, Monitoring the performance of commercial t1-rate transmission service, IBM J. Res. Dev. 35 (5.6) (1991) 805–814.

[28] B. Cornaglia, M. Spini, Letter: new statistical model for burst error distribution, Eur. Trans. Telecommun. 7 (3) (1996) 267–272.

[29] D.S. Berger, F. Gringoli, N. Facchi, I. Martinovic, J. Schmitt, Gaining insight on friendly jamming in a real-world IEEE 802.11 network, in: Proceedings of the 2014 ACM Conference on Security and Privacy in Wireless & Mobile Networks, in: WiSec '14, ACM, New York, NY, USA, 2014, pp. 105–116, doi:10.1145/2627393.2627403.

[30] IEEE standard for ethernet - Section 6, IEEE Std 802.3-2012 (Revision to IEEE Std 802.3-2008) (2012) 1–0. doi:10.1109/IEEESTD.2012.6419740.