Experience with an Implementation of the *Idle Sense* Wireless Access Method

Yan Grunenberger, Martin Heusse, Franck Rousseau, Andrzej Duda
*
LIG - Grenoble Informatics Laboratory
Grenoble, France
{ygrunenb,heusse,rousseau,duda}@imag.fr

ABSTRACT

An overwhelming part of research work on wireless networks validates new concepts or protocols with simulation or analytical modeling. Unlike this approach, we present our experience with implementing the *Idle Sense* access method on programmable off-the-shelf hardware—the Intel IPW2915/abg chipset. We also present measurements and performance comparisons of *Idle Sense* with respect to the Intel implementation of the 802.11 DCF (*Distributed Coordination Function*) standard.

Implementing a modified MAC protocol on constrained devices presents several challenges: difficulty of programming without support for multiplication, division, and floating point arithmetic, absence of support for debugging and high precision measurement. To achieve our objectives, we had to overcome the limitations of the hardware platform and solve several issues. In particular, we have implemented the adaptation algorithm with approximate values of control parameters without the division operation and taken advantage of some fields in data frames to trace the execution and test the implemented access method. Finally, we have measured its performance to confirm good properties of *Idle Sense*: it obtains slightly better throughput, much better fairness, and significantly lower collision rate compared to the Intel implementation of the 802.11 DCF standard.

Categories and Subject Descriptors

C.2.5 [**Computer-Communication Networks**]: Local and Wide-Area Networks—*Access schemes*

*LIG is a joint research laboratory of CNRS (*Centre National de la Recherche Scientifique*), INRIA (*Institut National de Recherche en Informatique et Automatique*), INP Grenoble (*Institut Polytechnique de Grenoble*), UJF (*Université Joseph Fourier*), and UPMF (*Université Pierre-Mendès-France*).

*CoNEXT'*07, December 10-13, 2007, New York, NY, U.S.A.

General Terms

Implementation, Performance, Measurements

Keywords

Wireless LANs, 802.11, Access Methods, Fairness

1. INTRODUCTION

The IEEE 802.11 standard for wireless LANs [20] has become an increasingly important technology with a wide adoption in various devices such as laptops, PDAs, handheld game consoles, TV set-top boxes, and even VoIP phones. 802.11 access points are extensively deployed to offer easy and tetherless access to the Internet in highly populated areas.

This success has also spawn an increased interest in research on improving performance and enhancing functionality of 802.11 wireless networks. For a given physical layer, its performance mainly depends on the access method to the common radio channel. The 802.11 DCF (*Distributed Coordination Function*) access method presents several drawbacks that limit its performance. If the number of contending stations is greater than the optimal (i.e. between 3 and 4 for 802.11b), stations experience a significant collision rate. Moreover, as a station cannot distinguish between collisions and failed transmissions, its performance lowers in imperfect channel conditions, because it reacts as if there were an increased contention between stations. DCF also leads to short term unfairness due to the *binary exponential backoff* in which a station doubles the contention window after a failed transmission: in this case, stations with an increased contention window suffer from lower channel access probability [2, 3]. Finally, DCF experiences a performance anomaly when contending stations use various bit rates at the physical layer [18].

A substantial amount of work has focused on tuning the operation of the 802.11 DCF (*Distributed Coordination Function*) access method, for instance improving contention window adaptation (*Slow Decrease* [32], *Asymptotically Optimal Backoff* [10]). Other authors

have even proposed new access schemes such as TCF (*TDM-based Coordination Function*) inspired by TDMA [26].

To maximize throughput and improve fairness, we have proposed the *Idle Sense* access method [19]. In *Idle Sense*, contending stations do not perform the exponential backoff algorithm after collisions or failed transmissions, rather they make their contention windows dynamically converge in a fully distributed way to similar values solely by tracking the number of idle slots between consecutive transmissions. By comparing the estimate with a theoretically derived value, stations adjust their contention windows CW using an AIMD (*Additive Increase Multiplicative Decrease*) control algorithm. In this way, the method decouples the dynamic load control from collision perception and approaches the optimal tradeoff between high throughput, low collision overhead, and good short-term fairness, which results in low delay. Unlike other proposals, *Idle Sense* enables each station to estimate its frame error rate, which can be used for switching to the right bit rate. Moreover, it solves the performance anomaly problem.

An overwhelming part of research work on wireless networks validates new concepts or protocols with simulation. However, even widely used simulators such as NS2, OPNET, or GloMoSim do not provide sufficiently realistic models of complex wireless behavior and their results may significantly differ for the same problem [15]. Experimental evaluation of wireless ad hoc networks also shows that much of the theoretical work on wireless protocols is based on wrong assumptions [24]. It is thus astonishing that there are only a few attempts to implement and measure proposed modifications to 802.11 wireless networks [17,25,30,31].

In this paper, we report on our experience with implementing *Idle Sense* on programmable off-the-shelf hardware. We present several families of wireless adapters considered for implementation and explain our choice as well as the tradeoffs we had to make during implementation. We have faced several problems raised by strong limitations in terms of memory and processor functionality (no multiplication, division, floating point arithmetic). As a result, the implemented algorithm is slightly different from the theoretical one. Debugging the code was extremely hard and tedious, so we used some fields in data frames to trace the execution and test the access method. Finally, we have performed several experiments to validate the implementation and verify that the cards with the implemented method conform to the specification. We then used them to measure performance and compare with the Intel implementation of the IEEE 802.11 DCF. Our measurements show a significant performance improvement with respect to DCF.

The paper is organized as follows. We start with the description of the 802.11 DCF and *Idle Sense* access methods 2. Section 3 overviews the related work. In Section 4, we compare several available wireless cards so to choose a suitable one for implementation. Section 5 reports on the implementation experience and Section 6 presents the measurements done using the cards with the implemented access method. Section 7 concludes the paper.

2. BACKGROUND

Before presenting our implementation of *Idle Sense*, we briefly review the details of 802.11 DCF and *Idle Sense* access methods.

2.1 802.11 DCF

802.11 DCF uses the *Carrier Sense Multiple Access/ Collision Avoidance* (CSMA/CA) principle: before initiating a transmission, a station senses the state of the channel. If the medium is sensed busy, the station waits until the channel is free during a *Distributed Interframe Space* (DIFS) interval, afterwards, it waits for an additional random contention time. The station chooses a backoff time that is an integer number of time slots distributed uniformly in the contention window $[0, CW-1]$; if another transmission occurs during this procedure, the residual backoff is kept for the next contention period. The value of CW is set to $CW_{\min}$ for the first transmission attempt and it is increased in integer powers of 2 at each failed transmission (collision or frame loss) up to $CW_{\max}$ (*exponential backoff mechanism*). If the destination correctly receives a frame, sends an ACK frame after a short period of time (SIFS). If the sending station does not receive the ACK frame after a certain delay, it assume that a collision occurred and tries to retransmit the frame.

In this mode of operation, when several stations contend for the channel, the one with the smallest backoff wins channel access. This value results either from a random value in its contention window or it is the remainder of a previously chosen backoff count down frozen by a transmission. If more than one station uses the smallest value, a collision occurs and stations apply the exponential backoff mechanism. Otherwise, when a station starts transmitting, other stations suspend the count down of their backoff and resume access procedure after the end of the transmission.

2.2 Idle Sense

Idle Sense optimizes 802.11 DCF for high throughput and fairness: contending stations do not perform the exponential backoff algorithm after collisions or failed transmissions, rather they make their contention windows dynamically converge in a fully distributed way to similar values solely by tracking the number of idle slots between consecutive transmissions. The method

works as follows: each station measures n_i, the number of consecutive idle slots between two transmission attempts. Every *maxtrans* transmissions, it estimates $\hat{n_i}$, the average of observed values of n_i. Then, it uses $\hat{n_i}$ to adjust its contention window to the target value $n_{\mathrm{i}}^{\mathrm{target}}$ computed numerically for a given variant of IEEE 802.11 PHY and MAC parameters—its value is 5.68 for IEEE 802.11b and 3.91 for IEEE 802.11g [19]. When stations adjust their CW so that n_i converges to ${n_i}^{\mathrm{target}}$, their throughput is optimal.

The original *Idle Sense* algorithm [19] makes n_i converge to ${n_i}^{target}$ by applying AIMD (*Additive Increase Multiplicative Decrease*) [16] to transmission attempt probability P_{e}. When stations do not perform the exponential backoff mechanism after collision, the transmission attempt probability is as follows [8]:

$$P_{\mathrm{e}} = \frac{2}{CW + 1}. \quad (1)$$

If a station observes too many idle slots compared to the target, it needs to increase P_{e} additively, which in turn will decrease n_{i}, whereas if it observes too few idle slots, it needs to decrease P_{e} in a multiplicative way, which in turn will increase n_{i}. This leads to the following algorithm:

- If $n_i \geq {n_i}^{target}, P_{\mathrm{e}} \leftarrow P_{\mathrm{e}} + \epsilon$
- If $n_i < {n_i}^{target}, P_{\mathrm{e}} \leftarrow \alpha P_{\mathrm{e}}$

where ϵ and α are some adaptation parameters. In practice, stations update their CWs by combining these updating rules with Eq. 1.

Idle Sense proposes a distributed mechanism to control channel contention that solves all drawbacks of DCF. It allows to distinguish the part of failed transmissions due to collisions from the failed transmissions due to imperfect channel conditions. Hence, it is able to decouple the load control from transmission adaptation, and solve the problems addressed by bit rate control algorithms such as ARF [22]. We have studied *Idle Sense* analytically and by simulation in a single cell [19] as well as in a multi-cell environment [28].

3. RELATED WORK

In recent years, many authors have analyzed performance of the 802.11 DCF access method and proposed various modifications. Calì *et al.* have arrived at the conclusion that a suitable control algorithm of contention windows may lead DCF to its theoretical performance limits [12, 14]. Many proposal have considered replacing the exponential backoff, for instance Bharghavan *et al.* proposed MACAW [7] in which the frame header includes the current value of the backoff counter thus enabling all listening stations to use the same value. They also added the Multiplicative Increase Linear Decrease adaptation algorithm to prevent large oscillation of the backoff. Bensaou *et al.* proposed an algorithm to ensure fairness by maintaining a ratio between the achieved bandwidth and the required channel part [1]. Both proposals require ideal channel conditions for transmitting a control parameter between contending stations, which is difficult to obtain in a wireless environment.

Other approaches consider channel utilization to adapt contention windows. DCC (*Distributed Contention Control*) [9] requires stations to regularly compute a variable called *Slot_Utilization*, which is basically the ratio between the number of busy slots over available slots. Before a transmission, a station computes the probability of transmission attempt from *Slot_Utilization* and the previous value. The method lowers collision rate and enhances performance. By computing the performance of a *p-persistent* 802.11 access scheme [14] and then by designing a backoff tuning scheme [13], Calì *et al.* have showed how to determine the optimal *Slot_Utilization* by evaluating the number of competing stations. The Asymptotic Optimal Backoff (AOB) resulting from this work achieves performance near the theoretical optimal capacity of the IEEE 802.11 protocol [11]. However, AOB retains the exponential backoff mechanism of DCF, so it does not completely decouple collision detection from load control.

As mentioned in the introduction, only a few papers have dealt with the problem of implementing access methods for 802.11 wireless networks. Actually, such an approach to validating new proposals requires much effort and presents many difficulties. There is only a small number of possible options for doing this: (i) develop custom hardware and associated firmware, which allows for any modification, but requires a substantial development effort; (ii) use off-the-shelf hardware, which requires access to the firmware source code and a development environment provided by the manufacturer; (iii) use open source drivers like MadWifi [29], which does not give access to the real time parts of the code, so it is of very limited use for low level modifications.

Bernasconi *et al.* have implemented the AOB scheme according to the requirements similar to ours [5, 6]. The implementation follows the first option—it includes Radio Frequency up/down converters (RF), a Baseband Modulator/Demodulator (BB), and the MAC firmware running on a compact DSP micro-controller. The custom board has a JTAG[1] interface for debugging purposes and uploading the MAC firmware onto the on-board flash memory. This hardware gives access to the low level real-time operations of the MAC layer thus enabling a full implementation of a new MAC protocol. However, as stated by the authors, they still need the development board to conduct large scale experiments.

[1]Joint Test Action Group abusively used for the Test Access Port - TAP.

Di Stefano *et al.* have developed a hardware platform based again on custom-made FPGA wireless cards for precise measurements of 802.11 WLANs and reported on how commercially available cards perform and react to a non-standard behavior [33].

To our knowledge, only few papers report work in which the authors have tried to modify the inner workings of existing 802.11 cards. Lacage *et al.* have modified the ARF (*Automatic Rate Fallback*) adaptation mechanism using the open source MadWifi driver for Atheros chipsets [25]. They have used the host system to keep track of the success/failure of transmission for multiple rate, and implemented an algorithm as a kernel module on the host system. Ng *et al.* have evaluated TCP performance and fairness on a 802.11e testbed that implemented 802.11e EDCA in the MadWifi driver [31].

4. HARDWARE REQUIREMENTS

Our objective is to implement *Idle Sense* on a significant number of operational cards at a reasonable price and limited effort. Thus, we have decided to avoid the burden of developing custom hardware and preferred to modify the firmware of available off-the shelf wireless cards. The number of available chipsets that implement the IEEE 802.11 standard is fairly large, but they must fulfill some specific requirements: *Idle Sense* relies on the ability of monitoring the number of idle slots and controlling the contention window after each transmission.

Basically, we can divide the available off-the shelf wireless cards into several categories. The first type of the 802.11 wireless cards is hard wired. Such cards provide hard-coded logic to support the operation of the 802.11 MAC protocol. A station controls the chip through a limited set of commands and transfers frames over the PCI bus. Another class of cards provides a large set of interactions between the host system and the wireless chipset thus enabling a lot of flexibility limited only by the constraints of the host system. The last class of cards executes the code of the 802.11 MAC protocol on a specialized processor. Code can be uploaded on boot or upgraded by memory flashing.

Our implementation objectives have restricted the choice of a hardware platform to cards that provide access to specific functions: slot counting and contention window management. Research community widely uses Atheros chipsets, because they support the flexible MadWifi driver [29] that allows to modify some functions of the MAC layer. However, we cannot rely on Atheros chipsets to implement *Idle Sense*, because their open source API does not provide access to the function of sending frames. At best, we can obtain feedback on the last packet sent along with the information about a successful transmission, a success with retries, or a failure. Actually, the existence of queues implemented in hardware suggests that the cards themselves manage sending frames, which seems reasonable as the host system running a standard operating system like Linux or FreeBSD does not guarantee real-time constraints [25]. So, we had to focus on hardware platforms providing flexible low-level software.

Another possible candidate was the Prism54 chipset[2]. A host system uploads the firmware upon driver initialization so that the card can behave in different manners depending on the used firmware. It also supports the possibility of adding new features, for instance the capability to manage new types of frames. The Prism54 project[3] provides some interesting information on the inner workings of the chipset: it is basically an ARM device connected to a RF component. The project offers two types of firmware: FullMAC and SoftMAC. FullMAC is the original firmware for the first adapters developed by Intersil. It requires monolithic firmware implementing the 802.11 MAC to be loaded onto the device. To reduce production costs, SoftMAC firmware has been developed later by splitting FullMAC into two parts: the LMAC (Lower MAC) device firmware executed on the embedded ARM processor and the UMAC (Upper MAC) binary library executed on the host that provides an opaque API to the device driver. Reverse engineering on a specific part of the firmware has shown that sending frames is implemented in software, so this class of devices is suitable for implementing our access method. However, the main obstacle is the need for reverse engineering of code to be able to modify the required parts of the firmware.

Intel cards based either on the IPW2200 or IPW2915 chipsets[4] have the same kind of software architecture: the firmware controls the behavior of the device. As we can learn from the GPL driver available from Intel, the firmware is specific for each mode (infrastructure, ad hoc, or Access Point) and comes in three parts: bootcode, microcode, and firmware code. The firmware is uploaded by an open-source Linux/BSD driver onto the card before any operation. Afterwards, the driver sends commands to the device for performing required operations such as scan, association, transmission etc. Thanks to Intel (cf. Section 8), we have obtained access to the microcode part of the Intel chipset and could modify the required functions. The combination of hardware and software features offered by the Intel chipsets perfectly suited our objectives.

5. IMPLEMENTATION DETAILS

We present in this section the details of the *Idle Sense* implementation on Intel IPW2915 cards.

[2] `http://www.conexant.com/`
[3] `http://prism54.org/`
[4] `http://www.intel.com/network/connectivity/products/wireless/prowireless_mobile.htm`

5.1 Tuning the adaptation algorithm of Idle Sense

We have extensively tested and evaluated the original adaptation algorithm of *Idle Sense* [19]. We have observed that applying *AIMD* to P_e is not a right thing to do, in part because it requires a ceiling function to prevent P_e from becoming greater than 1, which reveals a shortcoming of the chosen approach. Adjusting P_e is straightforward considering channel contention theory, but then AIMD is probably not the right choice. In fact AIMD is more suitable for controlling a positive integer variable, because anyway multiplicative decrease keeps it positive and additive increase is not a problem. Thus, we propose to apply *AIMD* to the contention window, which yields the following algorithm:

- If $n_i \geq {n_i}^{target}$, $CW \leftarrow \alpha \cdot CW$
- If $n_i < {n_i}^{target}$, $CW \leftarrow CW + \epsilon$

The footprint of this updating scheme is also more compact than the previous one, as there is no need for maintaining P_e variable required for computing CW.

We have also observed that the accuracy of the algorithm improves, if we adjust the parameter *maxtrans* [19] so that it becomes proportional to the number of stations (recall that a station refreshes CW every *maxtrans* transmissions on the channel). Here the problem with the original algorithm was that when the number of stations increases, stations might update their CW several times between two of their transmissions, which is when CW is actually used. This means that for a given set of AIMD parameters, the behavior changes depending on the number of active stations, which is not satisfactory. So we use the fact that *Idle Sense* results in a CW value proportional to the number of stations in a cell. Then, to speed up convergence when $\hat{n}_i$ is clearly off target, we use a small value of *maxtrans* when n_i is significantly different from the target value. The refined adaptation mechanism is thus the following:

- If $\left|n_i - n_i^{target}\right| < \beta \rightarrow maxtrans = \frac{CW}{\gamma}$,
- If $\left|n_i - n_i^{target}\right| \geq \beta \rightarrow maxtrans = 5$,

where γ takes the value $\gamma = 4$. Using Eq. 5 and 9 in [19], this yields the following relations:

- for IEEE 802.11b, $maxtrans = \frac{CW}{\gamma} \approx 3N$
- for IEEE 802.11a/g, $maxtrans = \frac{CW}{\gamma} \approx 2N$

Algorithm 1 presents the formal specification of *Idle Sense*.

5.2 Simplified computations

So far, we have tested the performance of *Idle Sense* using simulation to tune different parameters such as

Algorithm 1 *Idle Sense*

```
maxtrans ← 5 ; sum ← 0 ; ntrans ← 0
After each transmission {
/* Station observes n_i idle slots before a transmission */
sum ← sum + n_i
ntrans ← ntrans + 1
if (ntrans >= maxtrans) then
  /* Compute the estimator */
  n̂_i ← sum/ntrans
  /* Reset variables */
  sum ← 0
  ntrans ← 0
  if (n̂_i < n_i^target) then
    /* Increase CW */
    CW ← CW + ε
  else
    /* Decrease CW */
    CW ← α · CW
  end if
  if (|n_i^target − n̂_i| < β) then
    maxtrans ← CW/γ
  else
    maxtrans ← 5
  end if
end if
}
```

n_i^{target} and the coefficients of AIMD control. We used two different discrete-event simulators: one that accurately models the PHY and MAC layers [27] and another one that only simulates the MAC layer [19]. The parameters were represented as floating point numbers, so all computations had good precision.

n_i^{target} is calculated from the optimal value of CW according to the formula of *Idle Sense* [19]. $\frac{1}{\alpha}$ and ϵ are the parameters of the AIMD algorithm while β and γ impact the stability of the feedback loop. The values of these parameters were tuned using simulations in order to obtain a good tradeoff between stability and convergence speed.

We have finally converged to the following values:

- $n_i^{\text{target}} = 3.91$
- $\frac{1}{\alpha} = 1.0666$
- $\epsilon = 6.0$
- $\beta = 0.75$
- $\gamma = 4$

As we have to implement *Idle Sense* on a processor that only provides integer arithmetic and does not support multiplication nor division operators, we need to carefully choose the right values and program computations in a right way.

We have decided to use the values that are close to the nearest power of 2 of a given parameter. For example, we can approximate α parameter, which is equal to 0.93755, with the ratio of $15/16 = 0.9375$ that can be computed as $1-1/16$ easy to do on any hardware (division by 2 corresponds to a register shift). We thus have adopted the following values:

- $n_{\mathrm{i}}^{\mathrm{target}} = 4$
- $\alpha = 1 - 1/16$
- $\epsilon = 6$
- $\beta = 1$
- $\gamma = 4$

```
; after each transmission:
        mov     ntrans AX
        addi    1 AX
        mov     AX ntrans
        mov     maxtrans BX
        cmp
        jmp     lt noupdate
; we have reached maximal transmission
        swap    AX BX
        shl     ; *2
        shl     ; *2
; we got maxtrans* target (=4)
        mov     sum BX
        cmp
        jmp     gt increase
decrease:
        mov     CW AX
        mov     AX BX
        shr     ; /2
        shr     ; /4
        shr     ; /8
        shr     ; /16
        swap    AX BX
        sub
; now we have 15/16 of CW in AX
        jmp     update
increase:
        mov     CW AX
        mov     6 BX
        add
update:
        mov     AX CW
noupdate:
```

Figure 1: Implemented *Idle Sense* access method in pseudo assembler code.

We can now rewrite our main control loop in pseudo assembler code using the chosen values of parameters (cf. Figure 1).

5.3 Generating Pseudo Random Numbers

In the IEEE 802.11 standard, the size of contention window CW only takes power of two values, for instance with $CW_{\min} = 31$ for 802.11b, a station chooses a uniformly distributed value between 0 and 31. The exponential backoff mechanism doubles the size of CW at each failed transmission up to the limit of $CW_{\max}$, which is equal to 1023 for 802.11b. Implementing backoff generation is thus straightforward: a station generates a pseudo random value in a k-bit register (for instance $k = 16$) and masks the required number of high order bits (for instance 11 bits to obtain a value between 0 and 31).

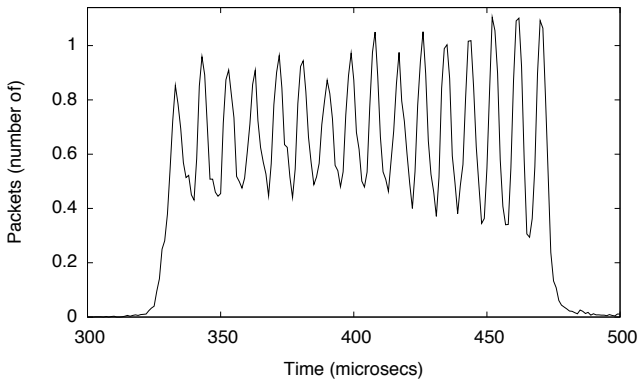

Figure 2: Interarrival histogram of the standard Intel firmware for 802.11a, $CW_{\min} = 15$.

Figure 2 presents an example of measuring the distribution of random values on the Intel cards for contention windows in 802.11a ($CW_{\min} = 15$). In the experiment, one station tries to send frames as fast as possible. We use *Interarrival Histograms*, a method for precise measurements (of the order of 1 μs) on standard hardware [4]. It is simple, yet powerful: a test wireless station sends a flow of packets to a wired host connected to a 802.11 access point via a direct 100 Mb/s Ethernet link. We record the timestamp of the packet arrival with the precision of 1 μs and compute the interarrival interval. The results are then presented as a histogram, which allows us to visually grasp the most frequent values of the transmission time and easily deduce some low-level characteristics of the tested wireless card. In the presented figure, we can easily observe 16 peaks of almost the same height corresponding to all the values in the contention window.

In *Idle Sense*, contention window CW may take any value, not necessarily a power of two values. This means that we need to find a different way of generating random values in the range $[0, CW]$. Our idea to solve the problem is to multiply CW (using a looped ADD operation) by a pseudo random number on k bits and then divide it by 2^k (using a shift operation). Algorithm 2 presents the implemented generation algorithm of random backoff in the range $[0, CW]$ (LFSR stands

Algorithm 2 *Generation of Random Backoff*

/* take upper 8 bits */
$random \leftarrow LFSR$ & $0x00FF$
$random >> 8$
$temp \leftarrow random * CW$
/* take upper 8 bits */
$backoff \leftarrow temp$ & $0xFF00$
$backoff >> 8$
}

Variable	Value	Comment
Random register	**10010011** 10101110	generated
Random register	00000000 **10010011**	register shift
CW	00000000 **00001101**	CW=13
Temp	**00000111** 01110111	Random * CW
Backoff	00000000 **00000111**	Backoff = 7

Figure 3: Example of random backoff generation for $CW = 13$.

for a Linear Feedback Shift Register used for generating pseudo random sequences). As the logical and arithmetic unit working on 16 bit words uses 8 bit long operands, the method is limited by the maximum value of $CW = 255$ on such hardware. This maximum value allows us to accommodate up to 28 active stations with performance gradually decreasing beyond this number. Incidentally, a more elaborate random generator could probably be added at an acceptable cost. An example of backoff generation with this method is presented in Figure 3.

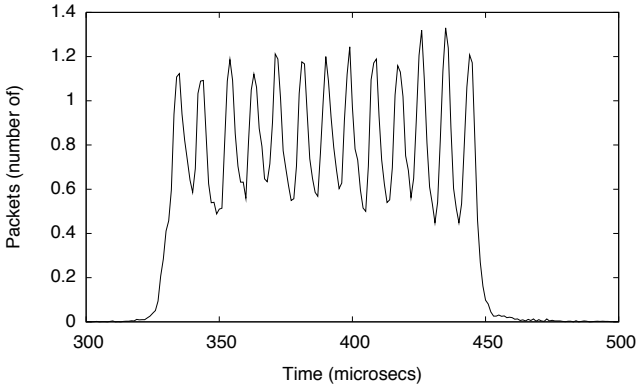

Figure 4: Interarrival histogram of the *Idle Sense* firmware with $CW = 12$.

Figures 4 and 5 present the histograms gathered in a similar experiment, but with the value of CW fixed to 12 and 13, respectively. We can observe fairly good statistical properties of the histograms showing that our generation method of random backoff follows the requirement of uniformly distributed values in a given range.

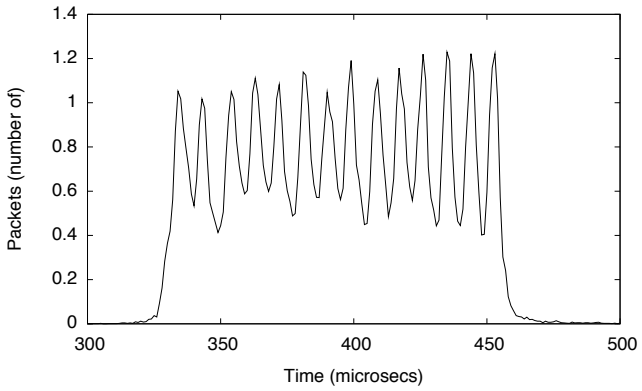

Figure 5: Interarrival histogram of the *Idle Sense* firmware with $CW = 13$.

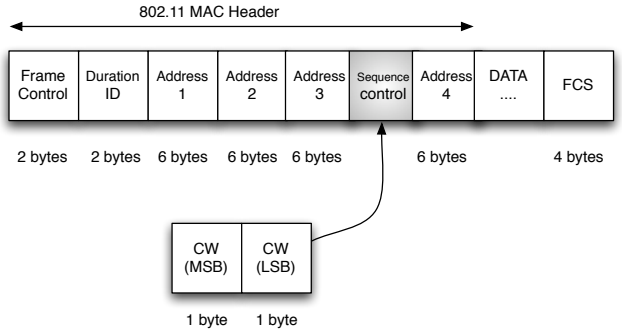

Figure 6: Use of the 802.11 header for debugging.

The resulting binary code of the implemented *Idle Sense* method and the random generator is slightly larger than the original code (excess of 64 Bytes over 16 KBytes).

5.4 Debugging

Debugging was one of the main problems in implementing *Idle Sense* on highly constrained hardware such as wireless cards. The limitations of processor capability and memory size have prevented us from using any form of execution tracing or controlled step-by-step execution. How to verify the right behavior of the contention window adaptation algorithm in such conditions? What we needed was a method for observing the current value of CW and evaluating the statistical properties of random backoff generation. As we used standard hardware, we could not obtain the value of CW during execution, because there is no JTAG port or any debug output. To solve this problem, we have decided to use communication for debugging purposes—we put the current value of CW into a 802.11 header field and capture all the frames sent by the card

under tests with a monitor station. We used the field of the sequence control number to display the contention window. Figure 6 presents the new usage of the 802.11 header.

Validating the correct operation of random backoff generation also required the use of the previously mentioned *Interarrival Histograms* method [4] to precisely measure time characteristics of the implemented MAC layer (cf. Figures 4 and 5).

6. MEASURED PERFORMANCE

In this section, we report on the measured performance of the *Idle Sense* firmware implemented on the Intel IPW2915 cards. We compare it to the standard 802.11 DCF firmware with respect to convergence speed, throughput, fairness, latency, and collision rate.

To illustrate their different behavior, we start by presenting how CW of stations evolves according to the number of active stations in both access methods. Then, we report on throughput measurements: we do not expect significant improvement here, because the throughput of DCF is near optimal for the number of stations we use in measurements. Even though their measured throughputs are similar, they are obtained through two contrasting access policies. Finally, we observe much better fairness, shorter delays, and less collisions of *Idle Sense* as predicted by simulations. When *Idle Sense* confronts DCF, it is less aggressive, so it obtains somehow lower throughput.

6.1 Testbed

We have set up an evaluation platform out of off-the-shelf standard Intel-based Centrino laptops. Our IPW2915 based cards operate in the relatively interference-free 5Ghz ISM band.

Most of the Linux and FreeBSD kernels can use our binary microcode provided that the microcode version is the same as the firmware and the driver (we used version 2.4). We used FreeBSD as the main operating system for generating traffic. An additional laptop with an Atheros card monitors traffic to capture debugging information in the frame header. Figure 7 presents the platform set up. We have included a dummynet link to emulate long link latency useful for measuring Internet access type of workload. The access point only serves to receive traffic generated by laptops towards the wired host. In this way, we only measure contention between laptops without any artifact that may be introduced when using two way traffic involving the access point.

6.2 Convergence Speed

Our first experiment concerned the evolution of contention windows in a dynamic scenario with varying number of active stations: a station becomes active each one second and starts generating UDP traffic.

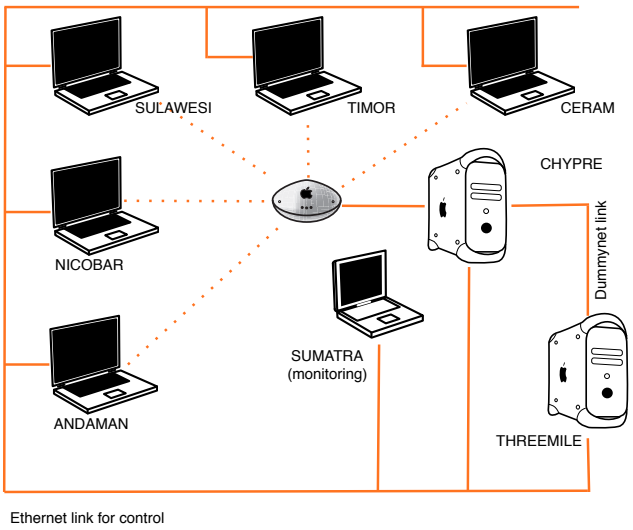

Figure 7: Testbed used for experiments.

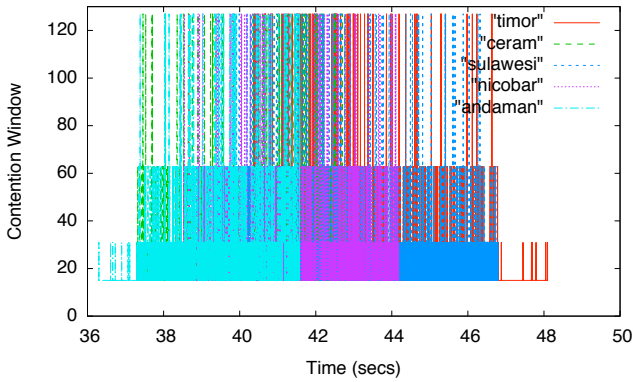

Figure 8: Evolution of the contention window for 802.11 DCF.

Figure 8 shows the evolution of the contention window for 802.11 DCF. At the beginning, CW is equal to $CW_{\min} = 15$ so the random backoff takes values between 0 and 15. Then, at instant 1s a second station starts contending for channel access. We can see a gap each time there is a retransmission, after which the value of CW doubles going from 15 to 31, 63, up to 127. After 5 seconds, five stations contend for the channel and the value of CW often stays around 127.

Figure 9 presents the behavior of *Idle Sense* in the same scenario. We can clearly identify different convergence phases after the increase of the number of contending stations. Upon the arrival of a new station, all stations converge to a new value of contention window, which is almost equal for all stations. The value attained for five stations is much smaller than those observed for 802.11 DCF.

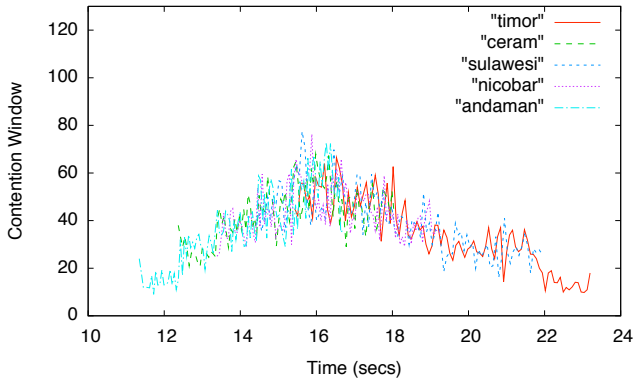

Figure 9: Evolution of the contention window for *Idle Sense*.

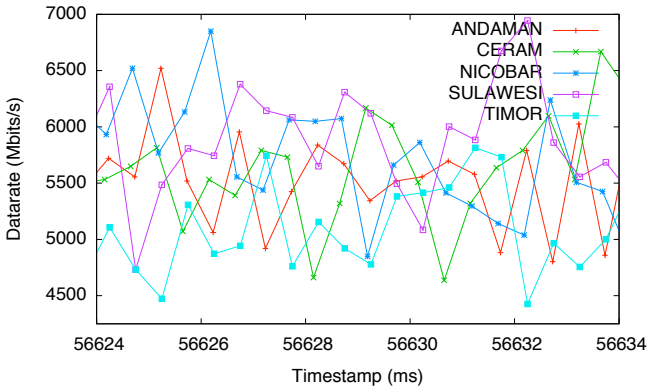

Figure 11: Throughput of five stations under 802.11 DCF.

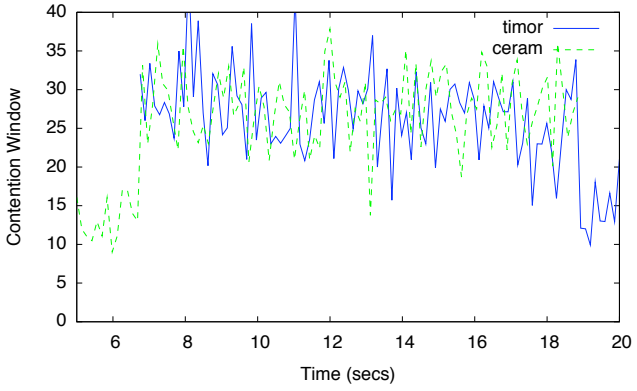

Figure 10: Convergence speed of the *Idle Sense* firmware.

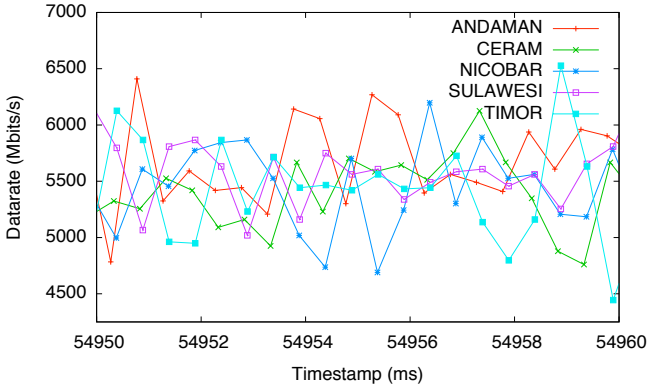

Figure 12: Throughput of five stations under *Idle Sense*.

Next, we wanted to examine the convergence speed of *Idle Sense* more precisely. At the beginning, one station generates traffic and at instant 2 s, the second station starts sending frames. Figure 10 shows that the AIMD adaptation algorithm enables the *Idle Sense* firmware to quickly adapt to varying network conditions as predicted by simulations.

6.3 Throughput

We have measured throughput obtained by the 802.11 DCF and *Idle Sense* firmware in a set up with five stations that try to send UDP data as fast as they can via the access point to a wired host. Stations use the bit rate of 54 Mb/s and the 5 GHz band. Using simulation, we obtained the following mean throughput: 5.848 Mb/s for DCF and 5.985 Mb/s for *Idle Sense*.

Figures 11 and 12 present the measured throughput obtained by contending stations for different access methods. We can see that the overall level of the throughput is similar for both access methods (total measured throughput is 28.3 Mb/s for DCF and 28.6 Mb/s for *Idle Sense*), but 802.11 DCF results in a much higher variance in throughput of different stations. For instance, TIMOR station obtains smaller throughput (5.46 Mb/s) compared to SULAWESI (5.9 Mb/s), because of short term unfairness. For the *Idle Sense* firmware, stations obtain throughput with a lower variance (5.9 Mb/s to 6.1 Mb/s), which results from better short term fairness. As we can see, experimental results and simulation results are consistent (DCF: 5.46<5.848<5.9 and *Idle Sense*: 5.9 <5.985<6.1).

6.4 Fairness

To evaluate the expected improvement in short term fairness when using the *Idle Sense* firmware, we did the following experiment: we set up five contending stations and monitored the arrival of each frame at the access point. Then, using the sliding window method, we computed the average Jain fairness index [21, 23] for a window of an increasing size. We normalize the window size so it is a multiple of the number

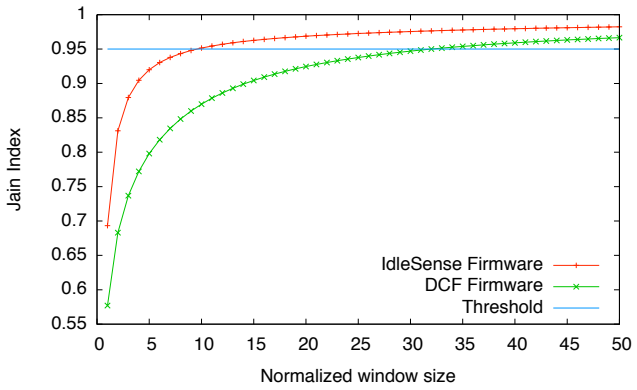

Figure 13: Jain index for 802.11 DCF and *Idle Sense* firmware, five contending stations.

of contending stations. The index of 1 represents perfect fairness, which is the case for instance of TDMA. Figure 13 compares the Jain index computed based on the measured values for both access methods. We can see that the threshold value of 0.95 (observed in original 802.11 experiments reported elsewhere [3]) is reached for a smaller window size with the *Idle Sense* firmware providing much better short term fairness than the 802.11 DCF.

6.5 Transmission statistics

We tried to evaluate the behavior of *Idle Sense* with respect to frame retransmissions. For this purpose, we monitored the transmissions of five contending stations and recorded frames. Using the 802.11 sequence number field as well as a special sequence number in the UDP payload, we isolated the frames successfully transmitted at the first attempt and the retransmitted frames (marked with the retransmit flag in the 802.11 header). As *Idle Sense* firmware enables stations to use a nearly optimal CW for a given number of contending stations, there should be less collisions and less wasted time in retransmissions. All stations wait for channel slightly more (recall that they try to set CW so that the average number of idle slots before a transmission is the same for all stations). In the 802.11 DCF, stations wait a little bit less, but after a collision the colliding stations wait much more time because of the contention window increased by the exponential backoff mechanism. Table 14 presents the results for 100000 captured frames, while Table 15 shows the statistics gathered during a 10 s session.

The results confirm the predicted properties of the *Idle Sense* firmware: DCF results in more retransmitted frames than *Idle Sense*. We have also observed that the *Idle Sense* cards transmit more frames at the first attempt than in the case of DCF. These statistics also explain the good throughput results presented in

Number of	802.11 DCF	*Idle Sense*
data frames	34086	40704
retransmissions	14253	7860

Figure 14: Transmission statistics for 100000 captured frames.

Number of	802.11 DCF	*Idle Sense*
data frames	17533	19505
retransmissions	7208	3707

Figure 15: Transmission statistics during a 10 s session.

Section 6.3: with *Idle Sense*, a station waits longer for transmitting a frame, but the probability of a successful transmission at a first attempt is greater. This tradeoff enables the *Idle Sense* stations to gain on short-term fairness as seen earlier.

6.6 Latency

By reducing collision probability, *Idle Sense* lowers the high variability of the interval between two channel accesses, which is the cause of the high latency variation in 802.11 networks. With *Idle Sense*, we expect to obtain more deterministic latency. To evaluate latency, we used four contending stations with UDP uploads and one station sending ICMP Echo request packets of 1472 bytes every 0.2 s for a total duration of 1 minute. Figure 16 presents the histogram of the ICMP packet latency. As we can see, there is no ICMP packet with latency over 15 ms with *Idle Sense*. The histogram is more equally distributed between 0 and 10 ms with *Idle Sense* than for DCF.

6.7 Interoperability

There is no fundamental incompatibility between DCF and *Idle Sense* access schemes. However, the design of *Idle Sense* is based on the assumption that each station aims at the same value of the target contention

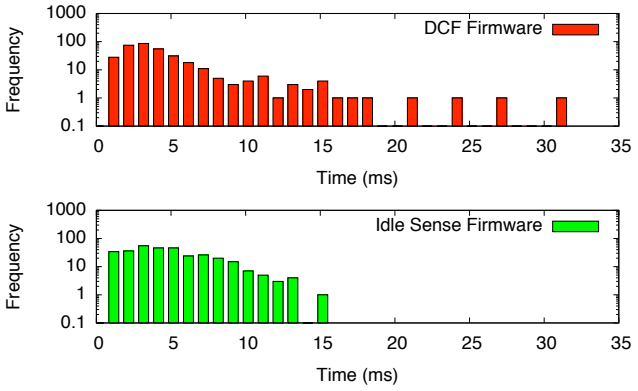

Figure 16: Latency histogram of ICMP packets during four concurrent UDP uploads.

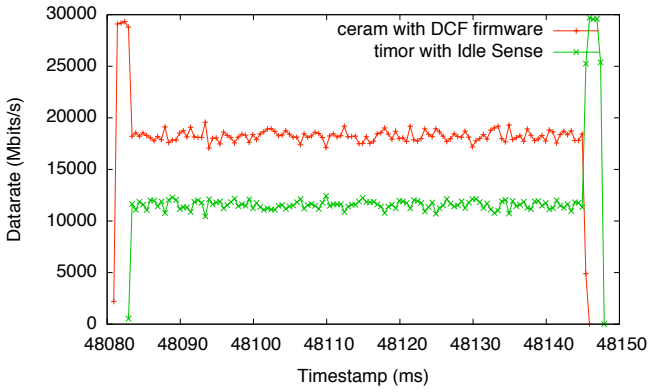

Figure 17: Throughput of contending DCF and *Idle Sense* stations.

window, so all stations need to use the same access method. When a network is composed by mixed DCF and *Idle Sense* stations, those that are less aggressive with respect to channel access will suffer from lower performance. To evaluate this effect, we have measured the throughput of two contending stations that use different firmware (cf. Figure 17). As expected, the *Idle Sense* firmware is less aggressive and leaves more access opportunity to DCF.

7. CONCLUSION

In this paper, we have presented our experience with an implementation of *Idle Sense* on programmable off-the-shelf hardware. We have started with tuning of the original algorithm for adapting contention windows. Then, we have analyzed different hardware platforms for implementing *Idle Sense*. Implementing a modified MAC protocol on constrained devices presents several challenges: difficulty of programming without support for multiplication, division, and floating point arithmetic, absence of support for debugging and high precision measurement. To achieve our objectives, we had to overcome the limitations of the hardware platform and solve several issues. In particular, we have implemented the adaptation algorithm with approximate values of control parameters without the division operation and taken advantage of some fields in data frames to trace the execution and test the implemented access method.

Our implementation has allowed us to extensively evaluate the performance of *Idle Sense* and compare it with the standard 802.11 DCF access method. Our measurements confirm good properties of *Idle Sense*: it obtains slightly better throughput, much better fairness, and significantly lower collision rate. Our measurements gathered on an experimental testbed provide a new insight in the domain in which paper design is prevalent—most of the proposed protocols are only validated by means of simulation or analytical modeling.

8. ACKNOWLEDGEMENT

The authors would like to thank the Intel Research Lab in Cambridge, UK, and Dina Papagiannaki in particular, for making the implementation of *IdleSense* possible. This work was partially supported by the European Commission project WIP under contract 27402 and the French Ministry of Research ANR project AIRNET under contract ANR-05-RNRT-012-01.

9. REFERENCES

[1] B. Bensaou, Y. Wang, and C. C. Ko. Fair Medium Access in 802.11 Based Wireless Ad-Hoc Networks. In *Proc. of MobiHoc'00*, Boston, MA, 2000.

[2] G. Berger-Sabbatel, A. Duda, O. Gaudouin, M. Heusse, and F. Rousseau. Fairness and its Impact on Delay in 802.11 Networks. In *Proc. of IEEE GLOBECOM 2004*, Dallas, USA, Nov. 29–Dec. 3, 2004.

[3] G. Berger-Sabbatel, A. Duda, M. Heusse, and F. Rousseau. Short-Term Fairness of 802.11 Networks with Several Hosts. In *Proc. of the Sixth IFIP IEEE International Conference on Mobile and Wireless Communication Networks, MWCN 2004*, Paris, France, Oct. 25–27, 2004.

[4] G. Berger-Sabbatel, Y. Grunenberger, M. Heusse, F. Rousseau, and A. Duda. Interarrival Histograms: A Method for Measuring Transmission Delays in 802.11 WLANs. Submitted for publication, `http://drakkar.imag.fr/spip.php?article242`, 2007.

[5] R. Bernasconi, R. Bruno, I. Defilippis, S. Giordano, and A. Puiatti. Experiments with an Enhanced MAC Architecture for Multi-hop Wireless Networks. In *Proc. of REALMAN'05*, July 2005.

[6] R. Bernasconi, S. Giordano, A. Puiatti, R. Bruno, and E. Gregori. Design and Implementation of an Enhanced 802.11 MAC Architecture for Single-Hop Wireless Networks. *EURASIP Journal on Wireless Communications and Networking*, 2007:Article ID 28315, 12 pages, 2007.

[7] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang. MACAW: a Media Access Protocol for Wireless LANs. In *Proc. of SIGCOMM'94*, pages 212–225, London, UK, 1994.

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

[9] L. Bononi, M. Conti, and L. Donatiello. Design and Performance Evaluation of a Distributed Contention Control (DCC) Mechanism for IEEE 802.11 Wireless Local Area Networks. In *Proc. of the 1st ACM International Workshop on Wireless*

Mobile Multimedia, 1998.

[10] L. Bononi, M. Conti, and E. Gregori. Runtime Optimization of IEEE 802.11 Wireless LANs Performance. *IEEE Transactions on Parallel and Distributed Systems*, 15(1):66– 80, January 2004.

[11] R. Bruno, M. Conti, and E. Gregori. Distributed Contention Control in Heterogeneous 802.11b WLANs. In *Proc. of WONS 2005*, 2005.

[12] F. Calì, M. Conti, and E. Gregori. IEEE 802.11 Wireless LAN: Capacity Analysis and Protocol Enhancement. In *Proc. of INFOCOM'98*, 1998.

[13] F. Calì, M. Conti, and E. Gregori. IEEE 802.11 protocol: design and performance evaluation of an adaptive backoff mechanism. *IEEE Journal on Selected Areas in Communications*, 18(9):1774–1786, 2000.

[14] F. Calì, M. Conti, and E. Gregori. Dynamic tuning of the IEEE 802.11 protocol. *ACM/IEEE Transactions on Networking*, 8(6):785–799, December 2000.

[15] D. Cavin, Y. Sasson, and A. Schiper. On the Accuracy of MANET Simulators. In *Proc of POMC '02, Second ACM International Workshop on Principles of Mobile Computing*, pages 38–43, 2002.

[16] D. Chiu and R. Jain. Analysis of the Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks. *Journal of Computer Networks and ISDN*, 17(1), June 1989.

[17] I. Dangerfield, D. Malone, and D. J. Leith. Testbed Evaluation of 802.11e EDCA for Enhanced Voice over WLAN Performance. In *Proc. of WiNMee, Second International Workshop on Wireless Network Measurement*, Boston, Massachusetts, USA, 2006.

[18] M. Heusse, F. Rousseau, G. Berger-Sabbatel, and A. Duda. Performance anomaly of 802.11b. In *Proc. of IEEE INFOCOM 2003*, San Francisco, USA, Mar. 30–Apr. 3, 2003.

[19] M. Heusse, F. Rousseau, R. Guillier, and A. Duda. Idle Sense: An Optimal Access Method for High Throughput and Fairness in Rate Diverse Wireless LANs. In *Proc. of ACM SIGCOMM 2005*, volume 35, pages 121–132, August 2005.

[20] IEEE. IEEE 802.11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, 1999.

[21] R. Jain, D. Chiu, and W. Hawe. A Quantitative Measure of Fairness and Discrimination for Resource Allocation in Shared Computer Systems. *DEC Research Report TR-301*, September 1984.

[22] N. Kim, S. Choi, and H. Yoon. IEEE 802.11b WLAN Performance with Variable Transmission Rates: In View of High Level Throughput. In *Proc. of the 4th International Conference on Networking, ICN 2005*, Reunion Island, France, Apr. 17–21, 2005.

[23] C. Koksal, H. Kassab, and H. Balakrishnan. An Analysis of Short-Term Fairness in Wireless Media Access Protocols. In *Proc. of ACM SIGMETRICS*, 2000.

[24] D. Kotz, C. Newport, R. S. Gray, J. Liu, Y. Yuan, and C. Elliott. Experimental Evaluation of Wireless Simulation Assumptions. Technical Report TR2004-507, Dept. of Computer Science, Dartmouth College, June 2004.

[25] M. Lacage, M. H. Manshaei, and T. Turletti. IEEE 802.11 Rate Adaptation: A Practical Approach. In *Proc. of the 7th ACM International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, 2004.

[26] C. Lim and C.-H. Choi. TDM-based Coordination Function(TCF) in WLAN for High Throughput. In *Proc. of GLOBECOM 2004*, Dallas, USA, November 2004.

[27] E. Lopez-Aguilera, J. Casademont, and J. Cotrina. Outdoor IEEE 802.11g Cellular Network Performance. In *Proc. of IEEE GLOBECOM 2004*, volume 5, pages 2992–2996, November-December 2004.

[28] E. López-Aguilera, M. Heusse, F. Rousseau, A. Duda, and J. Casademont. Performance of Wireless LAN Access Methods in Multicell Environments. In *Proc. of IEEE GLOBECOM 2006*, San Francisco, USA, Nov. 27–Dec. 1, 2006.

[29] MadWifi: Multiband Atheros Driver for WiFi. `http://madwifi.org/`.

[30] D. Malone, D. J. Leith, and I. Dangerfield. Verification of Common 802.11 MAC Model Assumptions. In *Proc. of PAM 2007, Passive and Active Measurement Conference, LNCS 4427*, pages 63–72, Louvain-la-neuve, Belgium, 2007.

[31] A. C. Ng, D. Malone, and D. J. Leith. Experimental Evaluation of TCP Performance and Fairness in an 802.11e Test-bed. In *Proc. of the ACM SIGCOMM E-WIND Workshop*, 2005.

[32] Q. Ni, I. Aad, C. Barakat, and T. Turletti. Modeling and Analysis of Slow CW Decrease for IEEE 802.11 WLAN. In *Proc. of PIMRC 2003*, 2003.

[33] A. D. Stefano, G. Terrazzino, L. Scalia, I. Tinnirello, G. Bianchi, and C. Giaconia. An Experimental Testbed and Methodology for Characterizing IEEE 802.11 Network Cards. In *Proc. of WOWMOM '06, IEEE International Symposium on a World of Wireless, Mobile, and Multimedia Networks*, pages 513–518, Niagara-Falls, Buffalo, NY, USA, 2006.