Energino: a Hardware and Software Solution for Energy Consumption Monitoring

Karina Gomez, Roberto Riggio, Tinku Rasheed, Daniele Miorandi, Fabrizio Granelli

▶ **To cite this version:**

Karina Gomez, Roberto Riggio, Tinku Rasheed, Daniele Miorandi, Fabrizio Granelli. Energino: a Hardware and Software Solution for Energy Consumption Monitoring. WiOpt'12: Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks, May 2012, Paderborn, Germany. pp.311-317. ⟨hal-00767059⟩

HAL Id: hal-00767059

https://inria.hal.science/hal-00767059v1

Submitted on 19 Dec 2012

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

HAL Authorization

Energino: a Hardware and Software Solution for Energy Consumption Monitoring

Karina Gomez†, Roberto Riggio†, Tinku Rasheed†, Daniele Miorandi†, Fabrizio Granelli‡
†CREATE-NET Research, Trento, Italy; *Email: first.secondname@create-net.org*
‡University of Trento, Trento, Italy; *Email: fabrizio.granelli@disi.unitn.it*

***Abstract*—Accurate measurement of energy consumption of practical wireless deployments is vital to the research community to develop pragmatic simulators and analytical models for the synthesis of new energy–aware and energy–efficient protocols and algorithms for wireless networks. However, there is considerable dearth in the availability of affordable and scalable energy consumption monitoring tools for the research community. In this paper, we introduce *Energino*, a scalable and affordable solution for energy consumption monitoring in wireless networks. The *Energino* power meter is a standalone plug–load meter based on the Arduino platform providing high resolution and sampling rate capabilities. We evaluate the capabilities and distinctive features of the *Energino* power meter in a real WLAN network. Results show that our solution is capable of isolating high resolution/frequency dynamics that can not be analyzed using commercially available tools. We also release both the hardware schematics and the software with a permissive license in order to encourage the research community to use and extend it.**

***Index Terms*—Energy consumption monitoring, hardware, software, wireless, testbed, arduino**

I. Introduction

Rationalizing energy consumption in ICT infrastructures is a growing concern for both industries and governments. As a matter of fact, several studies have shown that the ICT sector alone is responsible for an estimated 2% to 10% of the global energy consumption [1], [2], [3]. About 50% of this energy is actually consumed in wireless access networks [4], [5]. Such numbers should not come as a surprise especially if we consider the undeniable rise in the demand for mobile Internet access that we have witnessed in the last decade.

Partially fueled by the smartphones and Internet tablets revolution, Wireless LANs (WLANs) have found a second youth as a technology capable of relieving cellular network from the traffic burden of novel mobile applications. WLANs are nowadays extensively used by corporations, universities and municipalities in order to provide Internet connectivity to end users. Trends analysis reveals that the total amount of WLAN devices deployed has been increasing exponentially over the past few years [6]. In this context, optimizing the energy consumption of WLAN devices can significantly impact the overall CO_2 footprint of wireless networks [7].

However, while energy efficiency trade–offs have been taken into account for the users' terminals, which can be mobile or nomadic, less attention has been devoted to the network gateways, i.e. the Access Points, which are typically connected to the power grid and, thus, do not pose energy depletion challenges. As a result there is lack of best practices for designing energy efficient network protocols and architectures for broadband wireless access networks.

In this regard, simulator/emulator and analytical models are of capital importance in the design of new energy efficient protocols and algorithms for wireless networks. Nevertheless, energy consumption models used in mainstream tools such as OmNet++ [8] or ns3 [9] lack the level of details that is needed in order to steer research efforts towards the most energy critical aspect of the wireless systems. It is the authors' standpoint that such models must be based on an empirical assessment of the energy consumed by typical wireless devices. In order to pursue this approach two components are needed: (i) a reliable energy consumption measurement tool capable of identifying *where*, *when*, and *how* energy is consumed in a wireless network, and (ii) an effective methodology to characterize a wireless device's energy consumption behavior.

A preliminary work on the latter, i.e. a methodology to characterize energy consumption of wireless networks, has already been presented by the authors in [10], [11]. In this paper we introduce *Energino* a novel hardware and software solution for real–time energy consumption monitoring in wireless networks. The development of *Energino* was motivated by limitations, especially in term of sampling resolution and granularity, found in currently available commercial solutions for energy consumption monitoring. As a matter of fact, the device used in our previous work was unable to identify high resolution/frequency dynamics happening at the transition point between saturation and linear regimes. To the best of our knowledge, *Energino* is the first power meter capable of delivering high performance while remaining an affordable solution for large deployments.

The rest of the paper is structured as follows. Section II describes a number of requirements for an effective power consumption monitoring device and discuss how they have been turned into a set of design principles. In Sec. III the architecture of *Energino* is introduced. The reference use case exploited in order to refine our power consumption model together with the experimental methodology is summarized in Sec. IV. Experimental results are discussed in Sec. V. Finally, Sec. VI is devoted to the final conclusions and pointers to promising research directions.

This work was partially funded by the Research Project GREENET (PITNGA2010264759).

II. Requirements and design choices

The *Energino* meter has been designed around the following set of requirements:

- *High sampling rate*. Ideally, in order to isolate MAC–layer features such RTS/CTS handshakes and acknowledgments in WiFi networks a microsecond precision is needed. Therefore, if high frequencies dynamics, such as these MAC-level protocol transients are to be caught, an high frequency sampling rate is required.
- *High resolution*. In order to develop realistic models capable of capturing the correlation between different traffic patterns and energy consumption, it is mandatory to collect samples with a very fine granularity, ideally in the order of 10 mW or less.
- *Low cost/Low power*. The fraction of devices that can be actively monitored in a wireless network is clearly limited by cost to deploy *and* run such energy monitoring infrastructure. As a result, having a device that is both cheap to produce/assemble and that require very low power in order to operate is mandatory if a significant number of wireless devices are to be monitored.
- *Manageability*. Supporting basic management functions, such as being able to selectively turn on/off network devices or radio interfaces, enables the development of novel protocols and algorithms capable of adapting power consumption to the real network conditions (e.g., number of users and traffic patterns).
- *Autonomous*. Being able to operate as a stand–alone monitor without requiring tethering to either the device being monitored or to a third device is of capital importance in a highly distributed deployment. Such requirements mandate both a dedicated power supply and the support of networking functionality (e.g. Ethernet, WiFi, ZigBee).

Covering the entire spectrum of requirements with a single commercial product proved to be unfeasible. In particular, albeit several plug–loads meters are available as off–the–shelf solutions, they are all characterized by low frequency and low resolution sampling capabilities. Moreover, their prices range between 80€, for devices without network connectivity and very limited storage capabilities, up to 200€ for devices with full network connectivity (Ethernet or ZigBee).

The *Watts Up!* [12] meters are an example of devices belonging to this family. *Watts Up!* is a "plug load" meter that measures the amount of electricity used by whatever electrical appliance is plugged into it. Such measurements, taken with a granularity of 0.1W and a sampling period of 1s, can be logged into the device's internal memory or they can be exported using either an Ethernet port or a serial interface. Nevertheless, such specifications proved to be insufficient to catch dynamics occurring at the transition point between linear and saturation power consumption regimes [10], [11]. It is the authors' opinion that, a sampling period in the order of tens of times per second and a resolution of 10mW is required in order to properly investigate such high resolution/frequency power consumption dynamics.

High resolution and high frequency sampling can be easily obtained using a digital oscilloscope as done in [13]. However such a solution is expensive with a cost ranging from 500€ up to several thousand of euros. Moreover, meant serve as laboratory equipment, oscilloscopes are typically bulky and hard to integrate in an highly distributed network.

A summary of the advantages and disadvantages of both family of solutions is reported in Table I. As it can be seen, there is a clear space for novel energy consumption monitoring solutions. It is worth noticing that, albeit the design presented in this paper is primarily addressed towards the needs of the experimentally driven research, it can be easily extended to support also use cases coming from the industry such as monitoring of deployments powered using renewable sources and/or validation and characterization of the energy budget of metropolitan wireless networks.

III. Architecture

Energino is a plug–load meter designed to monitor the energy consumption of DC devices. It consists of an hardware and a software components both based on the Arduino platform. A management backend written in Python is used to configure *Energino*'s operating parameters, e.g. sampling rate and resolution, and to gather the energy consumption statistics. We release both the hardware schematics and the software with a permissive license[1] in order to encourage the research community to use and extend it.

A. Hardware

The need for a programmable and extensible platform drove us toward the Arduino platform. Arduino [14] is an open–source fast prototyping platform which, at its core, consists of a programmable microcontroller that can sense the environment using a variety of sensors and can affect its surroundings by controlling lights, motors, and other actuators. Additional modules, called "shields", can be used in order to extend the Arduino capabilities. Particularly relevant for our prototype are the extensions providing networking functionality (Ethernet, Wifi, and ZigBee).

The Arduino board supports 6 input channels using a 10-bit analog to digital converter. In particular, it maps input voltages between 0V and 5V to integer values between 0 and 1023. The ensuing resolution is thus 4.9 mV per unit. Each input can be sampled with a period of 100μs which results in a maximum sampling rate of 10kHz. Such a sampling rate is not high enough to catch MAC-level events such as acknowledgments however, we accepted it as a reasonable trade–off between cost/complexity and performance.

The Arduino has been extended with a custom module integrating a voltage sensor (based on a voltage divider), a current sensor (based on the Hall effect), and a solid state relay. A block diagram of the entire system is reported in Fig. 1. In the rest of this section we shall describe in details these three subsystems.

[1] Available at: http://www.wing-project.org/doku.php?id=energino

TABLE I: Approaches to power consumption monitoring.

	Resolution	*Sampling Rate*	*Price*	*Ease of deployment*
Plug-load meters	Low	Low	Average	High
Oscilloscopes	Very High	Very High	Very High	Low

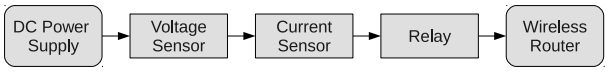


Fig. 1: Energino system architecture.

1) Voltage Sensor: The voltage sensor is implemented using a resistive voltage divider which produces an output voltage (V_{out}) that is a fraction of its input voltage (V_{in}). The divider consist of a series of two resistors R_1 and R_2. The relationship between input voltage, V_{in}, and output voltage, V_{out}, is given by $V_{out} = V_{in}R_2/(R_1 + R_2)$. The divider has been dimensioned to support input voltages up to 55V. The rationale behind this choice is that standard Power–Over–Ethernet injectors typically used to power networking devices are characterized by an output voltage of $48V$. In particular, $R_1 = 10k\Omega$ and $R_2 = 100k\Omega$, which result in a output voltage of 4.3V when an input voltage of 48V is applied. This is within the the Arduino constraints even if slight fluctuation of the input voltage occur.

2) Current Sensor: The current sensor consists of a linear Hall–effect[2] circuit capable of measuring currents between -5A and $+5$A. An applied current flowing through this circuit generates a magnetic field which is sensed by the integrated Hall IC and converted into a proportional voltage. The output of the device when no current is flowing trough the sensor is 2.5V. The sensitivity of the device is 185mV/A, i.e. for each ampere flowing trough the current sensor the voltage on its output linearly increases of 185mV. For example when a 5A current is applied, the corresponding voltage read on the output is 3.425V. Reversely, if a negative current of -5A is applied, the output will read 1.575V.

3) Relay: Finally, a solid state relay (SSR) is used in order to turn on/off the network device being monitored. An SSR is an electronic switching device where a small control signal controls a larger load. It comprises a voltage or current sensor which responds to an appropriate input (control signal), a solid–state electronic switching device which switches power to the load circuitry either on or off, and some coupling mechanism to enable the control signal to activate this switch without mechanical parts. The device used in our system is a Crydom MPDCD3-B[3]. This SSR relay can switch loads up to 3A@60VDC using a control voltage of 3 to 32 VDC.

B. Software

The Arduino hardware is programmed using the C/C++ programming language with some simplifications and modifications. A library, called *Wiring*, is provided in order to make common input/output operations easier. Finally, a cross–platform Integrated Development Environment (IDE) written in Java is made available. Such IDE supports basic editing capabilities (syntax highlighting, and automatic indentation) and is also capable of compiling and uploading programs to the board effectively supporting the entire application lifecycle.

The software which manages *Energino* periodically transmits over the USB interface the average power consumed by the monitored device during the last observation period. This is accomplished by: (i) fetching the outputs of the voltage (V_{raw}) and current (I_{raw}) sensors; (ii) converting them to, respectively, the actual voltage (V_{real}) and the actual current (I_{real}); and (iii) multiplying the two values. The actual input voltage is given by:

$$V_{real} = 0.0049V_{raw}\frac{R1 + R2}{R2} = 0.0293V_{raw}$$

Similarly, the actual input current is given by:

$$I_{real} = \frac{0.0049I_{raw} - 2.5}{0.185} = 0.026I_{raw} - 13.51$$

It is worth noticing that during each sampling period the Arduino continuously polls the voltage and the current sensors and accumulates the values into two separate registers. At the end of the polling period the average power consumption is computed and the result is sent over the USB interface. This is done in order to filter–out fluctuations in the values read from the analog inputs.

IV. Evaluation methodology

In this section we present the experimental evaluation of *Energino* in a practical scenario, namely: power consumption monitoring in an IEEE 802.11–based wireless network. The purpose of this evaluation is to improve the model derived in our previous works [10], [11] by taking into account high resolution/frequency dynamics happening at the transition point between saturation and linear power consumption regimes. Such analysis has not been possible in our previous work due to limitations in the power consumption monitor utilized. In particular, the 0.1W resolutions supported by the device was too coarse to catch dynamics that are actually occurring with a granularity of in the order of tens of mW.

A. Network setup

The testing environment is composed of a custom IEEE 802.11 Access Point (AP) and a single notebook acting as wireless client (see Fig. 2). The AP is part of the Berlin Open

[2]The Hall effect is the generation of an electric potential perpendicular to both an electric current flowing along a conducting material and an external magnetic field.

[3]Available at: http://www.crydom.com/en/Products/Catalog/m_p.pdf

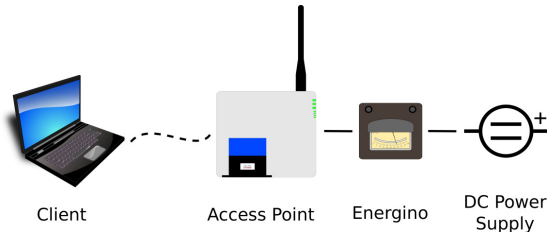


Fig. 2: Network setup used during our measurement campaign.

Wireless Lab (BOWL)[4] testbed deployed at Deutsche Telekom Laboratories in Berlin, Germany. BOWL [15] is a project of the Intelligent Networks (INET) group at Deutsche Telekom Laboratories. The main goal of the BOWL project is to provide an open platform for the wireless networking community.

The AP is built around a PCEngines ALIX 3D2 (500MHz x86 CPU, 256MB of RAM) processor board equipped with one IEEE 802.11n wireless interface with RTC/CTS disabled. The Access Point exploits OpenWRT 10.3.01-rc1 as operating system. The *ath9k* [16] Wireless NIC driver has been used during the measurements campaign. The AP's operating frequency was set to 5.12GHz (Channel 36). It is important to note that, unless otherwise specified, the rate adaptation algorithm has been set to *auto* and the transmission power has been left to its default value equal to 17dBm (∼50.12 mW) for all experiments. The notebook is a regular DELL Latitude 6420 equipped with an Intel PRO/Wireless 3945AB wireless adapter and running Ubuntu 10.04.

B. Power Consumption Monitoring

The power consumption statistics are collected at the *Access Point* using *Energino* with a granularity of 10 mW and a sampling period of 10ms. The sampling period can be set manually using the command line interface. The minimum sampling period supported by *Energino* is 1ms. It is important to remark that the power consumption is monitored for the whole device. Therefore, the results reported in this paper account for both the power consumed by the processing board for handling the incoming and outgoing traffic (e.g. for segmentation and reassembling, protocol overheads, computing checksums, etc.) as well as for the power consumed to deliver the actual frame over the wireless link (e.g. power amplifiers, modulator/demodulator, etc.).

C. Testing Methodology

The measurement campaign which we report in this paper aimed at assessing the actual power consumed by an IEEE 802.11 Access Point under different workloads. Traffic is injected at either the Access Point or the client and is consists of a single UDP flow. Power consumption measurements always refer to the Access Point. The power consumption of the Access Point in idle mode, i.e., without any data but the standard IEEE 802.11 beacons, has been measured as 4.2W.

[4]Available at: http://www.bowl.tu-berlin.de/

TABLE II: IEEE 802.11 OFDM Data Rates and Modulations

Modulation Type	Data Rate [Mb/s]
Binary Phase Shift Keying (BPSK)	6/9
Quadrature Phase Shift Keying (QPSK)	12/18
16-Quadrature Amplitude Modulation (16-QAM)	24/36
64-Quadrature Amplitude Modulation (64-QAM)	48/54

Traffic is generated using the Iperf traffic generator [17]. Results reported in this section are the average of measurements collected during 200 seconds with a 95% confidence interval. The following scenarios have been considered:

- *Variable Traffic Generation Rate, Fixed Datagram Length*. The datagram size is kept constant at 1000 bytes, while the traffic generation rate is progressively increased from 5Mb/s up to 55Mb/s in steps of 5Mb/s.
- *Variable Traffic Generation Rate, Fixed Modulation Type*. The datagram size is kept constant at 1000 bytes while the message generation rate is progressively increased. The rate control algorithm is disabled and the transmission rate is set manually using the command line interface. The experiment is repeated for each of the transmission rates supported by the wireless adapter (see Table II).
- *Constant Traffic Generation Rate, Variable Datagram Length*. The traffic generation rate is kept constant at 10Mb/s while the datagram size is progressively increased from 32 to 2048 bytes in steps of 256 bytes.

D. Experimental Measurements and Analysis

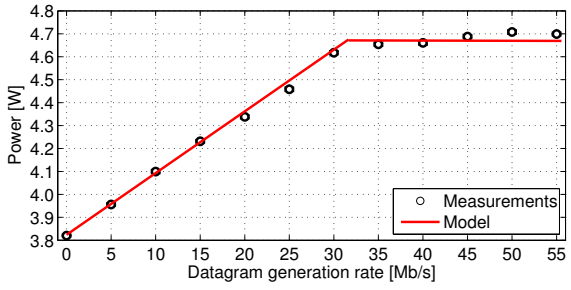
Figure 3 summarizes the behavior of the AP acting as either transmitter or receiver. The figure shows the average power consumption of the AP as a function of different traffic generation rates, for a fixed datagram size (1000 bytes). The packet loss is lower than 0.1%. We observe that:

(i) The overall power consumption behavior of the AP when it is acting as either transmitter and receiver is similar. However receiver mode is significantly more power efficient than transmitter mode.

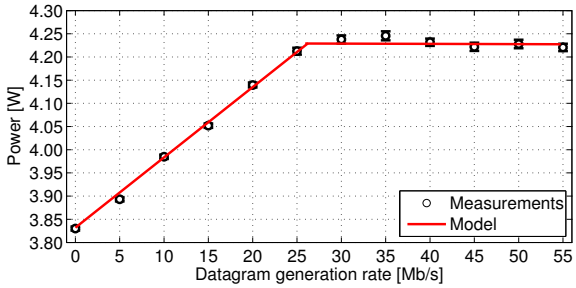
(ii) The power consumption is monotonically increasing with the traffic load until it reaches a saturation point. However, the saturation point and the power consumption at this point are different when the AP is acting as transmitter or receiver.

(iii) When the AP reaches the saturation point, the power consumption remains constant, as well as the maximum bandwidth. This is due to the saturation of the wireless interface when the offered traffic rate is higher than the physical link data–rate. Note that for an IEEE 802.11a link, the maximum MAC–layer data–rate is bounded to about 27Mpbs.

We next analyze how modulation and coding schemes impact power consumption. The results of this study are shown in Fig 4. We can observe that, lower modulation and coding schemes are more energy efficient than higher modulation and coding schemes when the AP is acting as transmitter. We can



(a) Access Point acting as transmitter.



(b) Access Point acting as receiver.

Fig. 3: Average power consumption (empirical and model) at the AP as a function of different traffic generation rates for a constant datagram size of 1000 bytes. Packet loss at the receiver has always been lower than 0.1%.

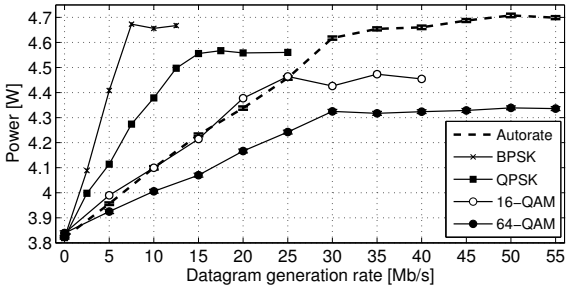


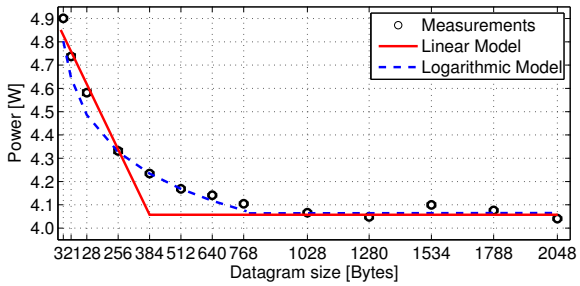
Fig. 4: Average power consumption at the AP as a function of different traffic generation rates for a constant datagram size of 1000 bytes for different modulation types and coding schemes. AP is acting as transmitter.

also observe that the power consumption level of the AP at the saturation point is different for each modulation.

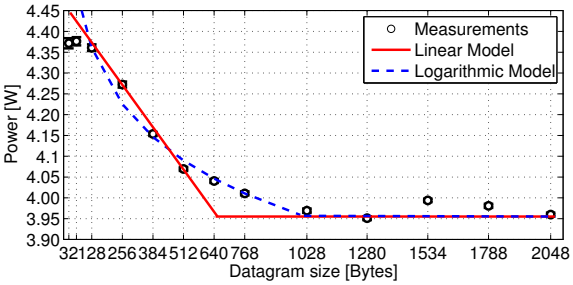
Finally, we analyze the impact of the datagram size on power consumption. As it can be seen from Fig. 5, the datagram size has a considerable impact on power consumption, in particular, we observe that:

(i) The AP consumes significantly more energy to transmit short frames. This result is intuitive and is related to MAC–layer protocol overheads (header and contention).

(i) When the datagram size is larger the the Maximum Transmission Unit of the link (which is typically set to 1500 bytes), the bandwidth utilization decreases and the power consumption increases. This behavior is due to (a) packet fragmentation and packet reassembling, (b) buffering and (c) MAC–layer overhead.



(a) Access Point acting as transmitter.



(b) Access Point acting as receiver.

Fig. 5: Average power consumption (empirical and model) at the AP as a function of the datagram size for a constant traffic generation rate of 10 Mb/s. Packet loss at the receiver side has always been lower than 1%.

We should finally stress that as the saturation point is reached, datagram loss increases due to buffer overflow at the transmitter side. Nevertheless, measurements carried–out at the receiver side have shown that packet loss due to link errors and/or collisions is always lower than 1%.

V. Enhanced Model

In this section, we validate the power consumption models derived in [10], [11] with the measurements obtained using the *Energino* power meter. The purpose of this validation is to highlight the limitations in resolution and sampling rate of the measurements carried out in our previous work using the *Watts Up!* power meter. Throughout the section, we use the following notation:

- x is the amount of traffic transmitted or received by the AP (expressed in Mb/s).
- s is the datagram size (expressed in bytes).
- $R.(x), S.(s)$, are the models which accounts for power consumption at the wireless gateway as function of, respectively, traffic rate and datagram size.

Notice that the notation $\cdot = Tx, Rx$ refer to the scenario when the wireless gateway is acting as transmitter and receiver, respectively. The $R.(x)$ and the $S.(s)$ models are:

$$R.(x) = \begin{cases} \alpha(s) \cdot x + \beta & \text{if } 0 \leq x \leq h(s) \text{ Mb/s,} \\ \gamma & \text{if } x > h(s) \text{ Mb/s,} \end{cases} \quad (1)$$

$$S.(s) = \begin{cases} -\delta(x) \cdot s + \epsilon(x) & \text{if } p \leq s \leq q \text{ byte,} \\ \eta(x) & \text{if } s > q \text{ byte,} \end{cases} \quad (2)$$

The parameters have the following physical meaning:

- $\alpha(s)$ $[\mu J/b]$ is the amount or energy spent by the wireless device in order to transmit or receive 1 bit from the session layer with a datagram size of s bytes;
- β $[W]$ is the amount of power consumed by the wireless gateway in idle mode;
- γ $[W]$ is the maximum amount of power consumed by the wireless gateway and represents the saturation power consumption;
- $\delta(x)$ $[\mu W/bytes]$ is the amount of power consumed by wireless gateway in order to transmit or receive 1 byte from the session layer arriving at a rate of x Mb/s;
- $\epsilon(x)$ $[W]$ is the maximum power consumed by the wireless gateway, transmitting at x Mb/s, using extremely small packets.
- $\eta(x)$ $[W]$ is the minimum power consumed by the wireless gateway to transmit traffic at a rate of x Mb/s.

A. R–Model Validation

In order to use our model to estimate the power consumption of a wireless device as a function of the transmission rate, we need three parameters:

(i) The amount of power consumed by the wireless access network gateway in idle mode (β). This value is easily obtained by measuring the power consumption of the wireless access network gateway without traffic.

(ii) The amount of power consumed by the wireless access network gateway in saturation regime (γ). This value is obtained by measuring the power consumption of the wireless access network gateway when an high amount of traffic is injected into the network. The rule of the thumb is to generate a traffic that is equal or higher than the wireless link's nominal data rate.

(iii) The amount of power consumed by the wireless access network gateway below the saturation regime (α). This value is obtained by measuring the power consumption of the wireless access network gateway when injecting an amount of traffic such that the packet loss over the wireless link is negligible (e.g. lower than 1%).

Table III reports the modeled parameters for R–Model obtained for a datagram size $s = 1000$ bytes. It is worth noticing that these values are obtained using only three datapoints from Fig. 3. Such data points have been chosen according to the above guidelines. The small RMSE proves the reliability of our methodology, see also Fig. 3a and Fig.3b where empirical energy consumption datapoints are plotted together with the values predicted using our model.

B. S-Model Validation

In order to use our model to estimate the power consumption of a wireless device as a function of the datagram size rate, we need three parameters:

TABLE III: R-Model parameters ($s = 1000$ bytes).

	$\alpha(s)$ $[\mu J/b]$	$\beta(s)$ $[W]$	γ $[W]$	$h(s)$ $[Mb/s]$	RMSE $[W]$
$f_{TX}(x)$	0.0259	3.8206	4.6543	32	0.0019
$f_{RX}(x)$	0.0155	3.83	4.2318	26	0.0001

TABLE IV: Linear S-Model parameters ($x = 10$ Mb/s).

	$\delta(x)$ $[\mu W/b]$	$\eta(x)$ $[W]$	$\epsilon(x)$ $[W]$	q q	RMSE $[W]$
$f_{TX}(s)$	-0.0022	4.900	4.066	384	0.0114
$f_{RX}(s)$	-0.00079	4.4751	3.9693	640	$3.9165 \cdot 10^{-4}$

(i) The amount of power consumed by the wireless access network gateway while transmitting/receiving at x Mb/s using the optimal datagram length ($\epsilon(x)$). This value is obtained by measuring the power consumption of the wireless gateway using as the datagram size the maximum transmission unit (MTU).

(ii) The amount of power consumed by the wireless access network gateway while transmitting/receiving at x Mb/s using a datagram length such that the packet loss over the wireless link is below 1% ($\eta(x)$).

(iii) The amount of power consumed by the wireless access network gateway while transmitting/receiving at x Mb/s using the null datagram length, i.e. no payload ($\eta(x)$).

Table IV reports the modeled parameters for the S–Model obtained for a rate $x = 10$Mb/s. As it can be seen from Fig. 5a and Fig. 5b the values predicted by our model do not match very well with the empirical data points. In particular, the power consumption behavior for datagram lengths between 256–768 bytes is clearly not linear.

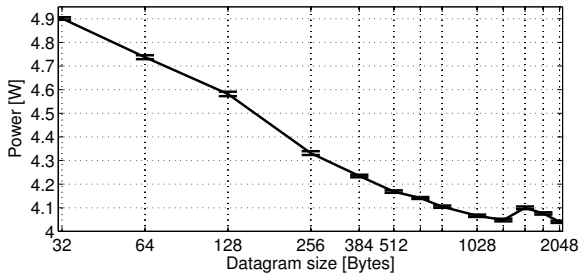
The discovery of this behavior has been made possible by the high sampling resolution supported by *Energino*. As a matter of fact, in our previous work [11], limitations in the *Watts Up!* power meter did not allow us to study power consumption dynamics with a granularity lower than 0.1W.

Starting from these insights, we observe that power consumption is actually linear when plotted in a semi–log scale as showed in Fig. 6a and Fig. 6b. Therefore, we decided to use the following logarithmic function for our model:

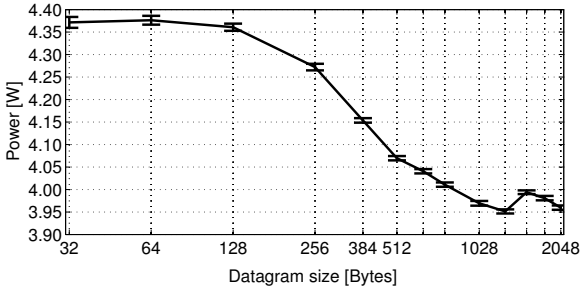
$$S.(s) = \begin{cases} -\delta(x) \cdot log(s) + \epsilon(x) & \text{if } p \leq s \leq q \text{ byte,} \\ \eta(x) & \text{if } s > q \text{ byte,} \end{cases} \quad (3)$$

Table V reports the modeled parameters for the new logarithmic S–Model obtained for a rate $x = 10$Mb/s. As it can be seen, the RMSE computed using the logarithmic model is significantly (one order of magnitude) lower than the RMSE obtained used the linear model.

It is worth noticing that, in order to fit the logarithmic model we used datapoints corresponding to datagram lengths greater than 128 bytes. The reason behind such choice is that due to MAC protocol overheads it is not possible to inject traffic at 10 Mb/s using payloads smaller than 128. As a matter of



(a) Access Point acting as transmitter.



(b) Access Point acting as receiver.

Fig. 6: Average power consumption at the AP as a function of the datagram size for a constant traffic generation rate of 10Mb/s plotted using a semi–logarithmic scale.

TABLE V: Logarithmic S-Model parameters ($x = 10$ Mb/s).

	$\delta(x)$ $[\mu W/b]$	$\eta(x)$ $[W]$	$\epsilon(x)$ $[W]$	q q	RMSE $[W]$
$f_{TX}(s)$	-0.2287	5.595	4.066	768	0.0091
$f_{RX}(s)$	-0.1953	5.3083	3.9693	1028	$8.0238 \cdot 10^{-5}$

fact injecting traffic at 10 Mb/s using a payload length of 32 bytes would require a packet injection rate of ≈ 39000 pkts/s, on the other hand the actual packet injection rate in a WiFi network has an upper bound mandated by the IEEE 802.11 Distributed Coordination Function (DCF) [18]. In particular it is easy to derive [19] that, using UDP packets that are 32 bytes long, the maximum number of packet transmission rate that can be sustained by an IEEE 802.11g interface using the highest modulation rate (54 Mb/s) is about 9800 pkts/s.

VI. Conclusions

In this paper, we introduced *Energino*, an affordable solution for real–time energy consumption monitoring in wireless networks. The hardware schematics and the software architecture were discussed in order to encourage the research community to use and extend *Energino*. We evaluated the distinctive features of our power meter in a realistic setting, namely an IEEE 802.11–based wireless network. As opposed to our previous work, based on a commercial solution, we could observe high resolution/frequency dynamics that take place at the transition between linear and saturation power consumption regimes. The ensuing empirical data allowed us to further refine our power consumption model for wireless access gateways.

As future work we plan to further investigate the power consumption of wireless access gateways when multiple users are sharing the wireless channel. Moreover we plan to combine the rate–based and the datagram length–based models into a single model capable of taking into account both aspects. Finally, on the power monitoring aspect we plan to enhance *Energino* by increasing the sampling resolution supported by the current sensor.

Acknowledgments

The authors would like to thank Dr. Cigdem Sengul for the valuable comments and discussions related to the power model and for allowing to use the BOWL Testbed infrastructure for the measurements.

References

[1] "Impacts of Information and Communication Technologies on Energy Efficiency," 2008. [Online]. Available: http://ec.europa.eu/information_society/activities/sustainable_growth/docs/studies/2008/2008_impact-of-ict_on_ee.pdf
[2] M. Marsan, L. Chiaraviglio, D. Ciullo, and M. Meo, "Optimal energy savings in cellular access networks," in *Proc. of IEEE ICC*, Dresden, Germany, 2009.
[3] K. Dufkova and, M. Bjelica, B. Moon, L. Kencl, and J.-Y. Le Boudec, "Energy savings for cellular network with evaluation of impact on data traffic performance," in *Proc. of IEEE EW*, Lucca, Italy, 2010.
[4] H.-O. Scheck, "ICT & Wireless Networks and their Impact on Global Warming," in *Prof. of IEEE EW*, Lucca, Italy, 2010.
[5] J. Lorincz, A. Capone, and M. Bogarelli, "Energy savings in wireless access networks through optimized network management," in *Proc. of the IEEE ISWPC*, Modena, Italy, 2010.
[6] A. P. Jardosh, K. Papagiannaki, E. M. Belding, K. C. Almeroth, G. Iannaccone, and B. Vinnakota, "Green WLANs: On-Demand WLAN Infrastructures," *Mob. Netw. Appl.*, vol. 14, pp. 798–814, December 2009.
[7] Y. Al-Hazmi, H. De Meer, K. A. Hummel, H. Meyer, M. Meo, and D. Remondo, "Energy-efficient wireless mesh infrastructures," *IEEE Network Magazine*, vol. 25, pp. 32–36, March-April 2011.
[8] "OMNeT++." [Online]. Available: http://www.omnetpp.org/
[9] "NS-3." [Online]. Available: http://www.nsnam.org/
[10] K. Gomez, R. Riggio, T. Rasheed, and F. Granelli, "Analysing the energy consumption behaviour of WiFi networks," in *Proc. of IEEE GreenCom*, Septmber 2011.
[11] K. Gomez, D. Boru, R. Riggio, T. Rasheed, D. Miorandi, and F. Granelli, "Measurement–based modelling of power consumption at wireless access network gateways," *Computer Networks*. [Online]. Available: http://disi.unitn.it/∼riggio/lib/exe/fetch.php?media=publications:greennets2012.pdf
[12] "Watts Up?" [Online]. Available: https://www.wattsupmeters.com/
[13] L. M. Feeney and M. Nilsson, "Investigating the energy consumption of a wireless network interface in an ad hoc networking environment," in *Proc. of the IEEE INFOCOM*, Anchorage, USA, April 2001.
[14] "Arduino." [Online]. Available: Available at: http://arduino.cc/
[15] T. Fischer, T. Huhn, R. Kuck, R. Merz, J. Schulz-Zander, and C. Sengul, "Experiences with BOWL: Managing an Outdoor WiFi Network (or How to Keep Both Internet Users and Researchers Happy?)," in *Proc. of USENIX LISA*, Boston, MA, USA, 2011.
[16] "Ath9k." [Online]. Available: http://linuxwireless.org/
[17] "Iperf." [Online]. Available: http://iperf.sourceforge.net/
[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," ANSI/IEEE, Jun. 2007.
[19] M. Gast, "When Is 54 Not Equal to 54? A Look at 802.11a, b, and g Throughput," 2003. [Online]. Available: http://www.oreillynet.com/pub/a/wireless/2003/08/08/wireless_throughput.html