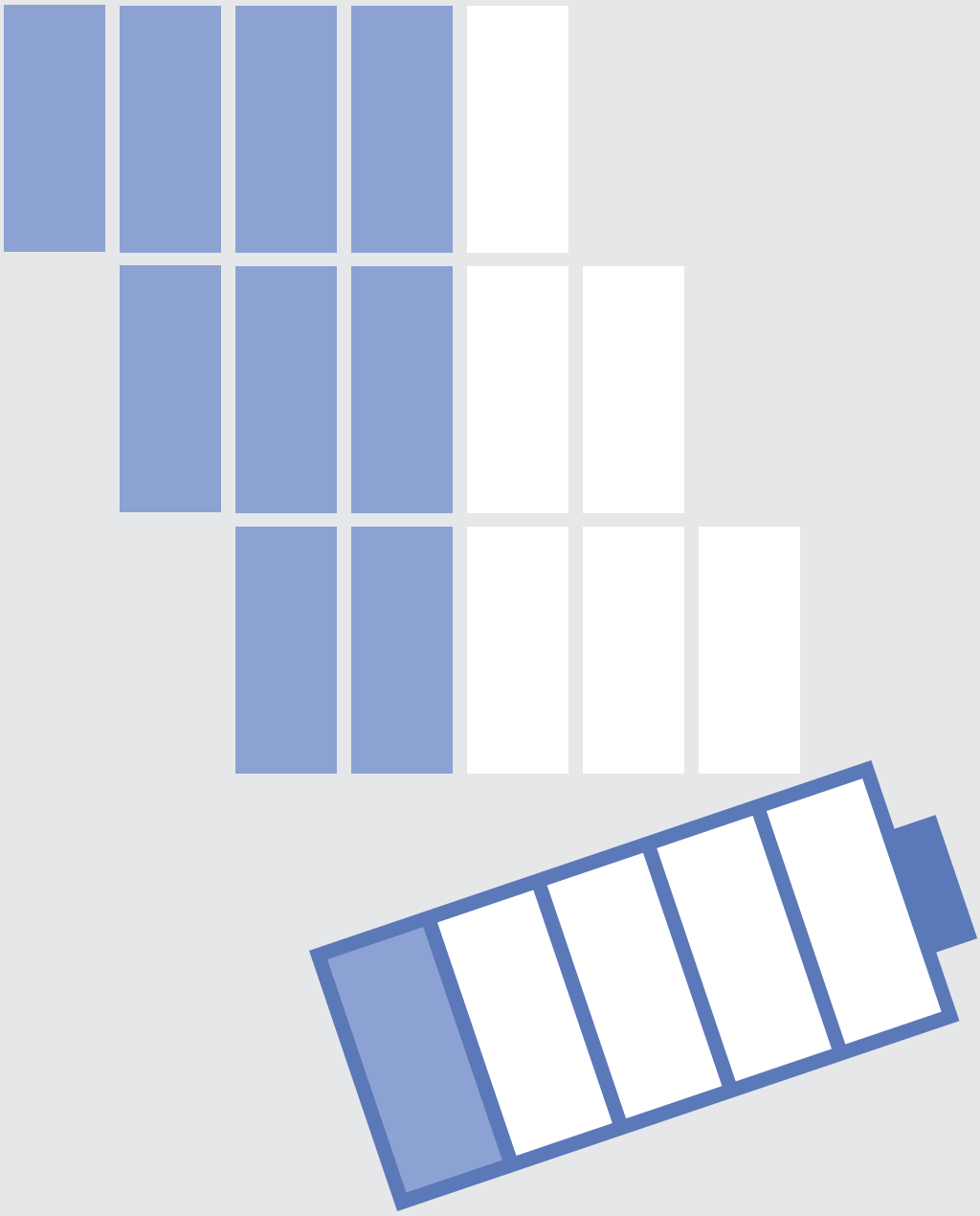


TESIS DOCTORAL

Energy Efficiency in Wireless Communications for Mobile User Devices

IÑAKI ÚCAR MARQUÉS

uc3m | Universidad Carlos III de Madrid

uc3m | Universidad Carlos III de Madrid

TESIS DOCTORAL

Energy Efficiency in Wireless Communications for Mobile User Devices

AUTOR: IÑAKI ÚCAR MARQUÉS

DIRECTOR: DR. ARTURO AZCORRA SALOÑA

Doctorado en Ingeniería Telemática · Leganés, mayo de 2018

PUBLISHED BY UNIVERSIDAD CARLOS III DE MADRID

First printing, May 2018

uc3m | Universidad Carlos III de Madrid

TESIS DOCTORAL

Energy Efficiency in Wireless Communications for Mobile User Devices

Autor: Iñaki Úcar Marqués
Director: Dr. Arturo Azcorra Saloña

Tribunal Calificador: Firma:

Presidente:

Vocal:

Secretario:

Calificación:

Leganés, a de de 2018

"Good scientific writing is not a matter of life and death;
it is much more serious than that."

—Robert A. Day, in *How to Write and Publish a Scientific Paper*.

Abstract

Mobile user devices' market has experienced an exponential growth worldwide over the last decade, and wireless communications are the main driver for the next generation of 5G networks. The ubiquity of battery-powered connected devices makes energy efficiency a major research issue.

While most studies assumed that network interfaces dominate the energy consumption of wireless communications, a recent work unveils that the frame processing carried out by the device could drain as much energy as the interface itself for many devices. This discovery poses doubts on prior energy models for wireless communications and forces us to reconsider existing energy-saving schemes.

From this standpoint, this thesis is devoted to the study of the energy efficiency of mobile user devices at multiple layers. To that end, we assemble a comprehensive energy measurement framework, and a robust methodology, to be able to characterise a wide range of mobile devices, as well as individual parts of such devices.

Building on this, we first delve into the energy consumption of frame processing within the devices' protocol stack. Our results identify the CPU as the leading cause of this energy consumption. Moreover, we discover that the characterisation of the energy toll ascribed to the device is much more complex than the previous work showed. Devices with complex

Resumen

El mercado de los dispositivos de usuario móviles ha experimentado un crecimiento exponencial a nivel mundial en la última década, y las comunicaciones inalámbricas son el principal motor de la siguiente generación de redes 5G. La ubicuidad de estos dispositivos alimentados por baterías hace de la eficiencia energética un importante tema de investigación.

Mientras muchos estudios asumían que la interfaz de red domina el consumo energético de las comunicaciones inalámbricas, un trabajo reciente revela que el procesado de tramas que se lleva a cabo en el dispositivo podría gastar tanta energía como la propia interfaz para muchos dispositivos. Este descubrimiento plantea dudas sobre los anteriores modelos energéticos para comunicaciones inalámbricas y nos obliga a reconsiderar los esquemas de ahorro energético existentes.

Desde este punto de vista, esta tesis está dedicada al estudio de la eficiencia energética de dispositivos de usuario móviles en múltiples capas. Para ello, se construye un completo sistema de medida de energía, y una metodología robusta, capaz de caracterizar un amplio rango de dispositivos móviles, así como partes individuales de tales dispositivos.

A partir de esto, en primer lugar se profundiza en el consumo energético del procesamiento de tramas en la pila de protocolos de los dispositivos. Nuestros resultados identifican a la CPU como principal causa de tal consumo. Además, se descubre que la caracterización de la cuota energética adscrita al dispositivo es mucho más compleja que lo mostrado por el trabajo anterior. Los dispositivos con CPU complejas (múltiples frecuencias y modos de apagado) requieren nuevas

CPUs (several frequencies and sleep states) require novel methodologies and models to successfully characterise their consumption.

We then turn our attention to lower levels of the communication stack by investigating the behaviour of idle WiFi interfaces. Due to the design of the 802.11 protocol, together with the growing trend of network densification, WiFi devices spend a long time receiving frames addressed to other devices when they might be dormant. In order to mitigate this issue, we study the timing constraints of a commercial WiFi card, which is developed into a standard-compliant algorithm that saves energy during such transmissions.

At a higher level, rate adaptation and power control techniques adapt data rate and output power to the channel conditions. However, these have been typically studied with other metrics rather than energy efficiency in mind (i.e., performance figures such as throughput and capacity). In fact, our analyses and simulations unveil an inherent trade-off between throughput and energy efficiency maximisation in 802.11. We show that rate adaptation and power control techniques may incur inefficiencies at mode transitions, and we provide energy-aware heuristics to make such decisions following a conservative approach.

Finally, our research experience on simulation methods pointed us towards the need for new simulation tools commited to the middle-way approach: less specificity than complex network simulators in exchange for easier and faster prototyping. As a result, we developed a process-oriented and trajectory-based discrete-event simulation package for the R language, which is designed as a easy-to-use yet powerful framework with automatic monitoring capabilities. The use of this simulator in networking is demonstrated through the energy modelling of an Internet-of-Things scenario with thousands of metering devices in just a few lines of code.

metodologías y modelos para caracterizar su consumo de manera existosa.

En este punto, volvemos nuestra atención hacia niveles más bajos de la pila de comunicaciones para investigar el comportamiento de las interfaces WiFi en estado inactivo. Debido al diseño del protocolo 802.11, junto con la tendencia creciente hacia la densificación de las redes, los dispositivos WiFi pasan mucho tiempo recibiendo tramas destinadas a otros dispositivos cuando podrían estar apagados. Para mitigar este problema, se estudian las limitaciones temporales de una tarjeta WiFi comercial, lo que posteriormente se utiliza para desarrollar un algoritmo conforme con el estándar que es capaz de ahorrar energía durante dichas transmisiones.

A un nivel más alto, las técnicas de adaptación de tasa y control de potencia adaptan la tasa de datos y la potencia de salida a las condiciones del canal. No obstante, estas técnicas han sido típicamente estudiadas con otras métricas en mente (i.e., figuras de rendimiento como la tasa total y la capacidad). De hecho, nuestros análisis y simulaciones desvelan un conflicto entre la maximización de la tasa total y la eficiencia energética en 802.11. Se muestra que las técnicas de adaptación de tasa y control de potencia pueden incurrir en ineficiencias en los cambios de modo, y se proporcionan heurísticos para tomar tales decisiones de un modo conservador y eficiente energéticamente.

Finalmente, nuestra experiencia investigadora en métodos de simulación nos hizo conscientes de la necesidad de nuevas herramientas de simulación comprometidas con un enfoque intermedio: menos especificidad que los complejos simuladores de redes a cambio de facilidad y rapidez en el prototipado. Como resultado, se desarrolló un paquete de simulación por eventos discretos para el lenguaje R orientado a procesos y basado en trayectorias, el cual está diseñado como una herramienta fácil de utilizar a la par que potente con capacidad de monitorización automática integrada. El uso de este simulador en redes se demuestra mediante el modelado en energía de un escenario de la Internet de las Cosas con miles de dispositivos de medida en tan solo unas pocas líneas de código.

Contents

List of Figures

Preface

Perhaps the reader reaches these words expecting a long diatribe about climate change and greenhouse emissions, about how we are wrecking our ecosystems. But I am *not* doing that. I am bored of it. I do not do it in the same way as a dissertation on astrophysics does not see fit to mention the fact that the Earth orbits the Sun: it should be obvious for everybody at this point. Instead, I will be appealing to a more selfish reason.

The preface of Matthew Gast's *802.11 Wireless Networks* summarises the tremendous success of wireless networks with just a couple of short bold statements:

> *People move. Networks don't.*

And although some researchers may be in a position to start to question the latter, I think that we all can agree on the first one. There is one small quibble though: *people move... for as long as their mobile devices allow them*. In fact, who has not experienced desperately looking for a socket in a cafeteria, a conference room, a metro station, a bus, an airport... because the phone is *dead*. As quarks inside a hadron[1], wireless devices have given us the freedom of mobility, but also the slavery of the battery.

[1] I know, I couldn't help it.

To sum up, we not only want *mobility*, but also *long-lasting* mobility. But the consumption of the wireless link is insignificant compared to the consumption of the screen, one might say. And it is true. However, if we for instance think of a smartphone, it is also true that it is in our pocket for most of the day, unused but still connected, managing instant notifications for multiple services. As a result, it is not uncommon to see the data connection as a top consumer in the phone's battery manager. So we can do better, and that is what this thesis is about.

I still remember the first time I met my advisor in a Skype interview. "What do you like the most: experimentation, mathematical

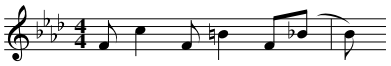
modelling or simulation?", it was one of the first things that he was interested to know. I replied that I like them all, but if I can[2] choose, I like experimental work the most. Because, for me, this is the *real* science; because, as Aristotle first reflected, *empiricism*, *careful observation*, is the starting point for *episteme*. In fact, the Spanish verb for "to see", which is *"ver"*, shares the Indo-European root **weid-* with the English adjective "wise": *I see, therefore I know*.

[2] Because sometimes you simply do not have the resources.

Still, I am a very curious person who likes to explore the many techniques available, and I think that this thesis is a faithful reflection of such curiosity. Thanks to this, during this journey I discovered myself as an open source enthusiast who enjoys making tools that are useful for other people. I even managed to grow a small community of users around one of such tools. Receiving feedback from them is extremely fulfilling, and for that I am very grateful.

But my curiosity is not limited to the method and extends to the research topics, as my publication record, including those prior to my PhD, inevitably evidences. The good thing about a PhD is that you have to focus on a specific topic; the bad thing about a PhD is that *you have to focus on a specific topic*. I have to admit that sometimes this has been a little burden to me which I hardly managed to overcome.

All in all, this has been a great experience. In fact, one of the best experiences of my life: I have met many interesting people from around the world, I have learned a lot from them and I have grown in every way as a person. Now this stage ends and an uncertain future opens up. But I am not afraid of it, because I am convinced that the best is yet to come.



Acknowledgements

I would like to begin by thanking my advisor, Arturo Azcorra, for giving me the opportunity of pursuing this PhD, and for being supportive and placing his trust in me even when I was beset by the *Publish-or-Perish* difficulties. His advice and guidance have been very valuable. I also wish to extend my thanks to UC3M's Department of Telematics Engineering, and to the NETCOM research group in particular, for supporting me. I have been privileged to work with and learn from great people.

I would also like to express my gratitude and sincere appreciation to all my co-authors, who always provide productive discussion, input and advice, and to all those who have helped me, shared their knowledge with me and guided my teaching experience.

My special thanks are due to Pablo Serrano, who in turn belongs to the latter two groups, and who has been like a second advisor to me; to Andrés García Saavedra, who is an outstanding engineer and

scientist, and whose guidance during the initial stage of my PhD was invaluable; to Ernesto García Ares, for his help and excellent work at UC3M's Technical Office in Electronics; to Bart Smeets, for his support, kindness and generosity. And of course to Francesco Gringoli, for having me during a great stay in Brescia. I was honored, and I really enjoyed a lot working closely with him; if I have not managed to fit part of that work in this dissertation, it is just my fault.

I am also indebted to so many others than would not fit in these pages. All my lab mates, past and present, and especially Luca, with whom I have shared the path since its very beginning, have contributed to a great working environment and have become my friends. Marco brought the seed of passion for tennis, which reached me through Fabio, and thanks to them I found an unparalleled stress reliever. Fernando's lessons and all my tennis mates have undoubtedly contributed to my sanity during this years.

And still, none of this would have been possible without the intangible contribution of a number of supportive relatives and friends. This thesis is especially dedicated to Almudena, who indulged me despite all the weekends and plans that were sacrificed to the contents of this book.

—Iñaki Úcar

Madrid, May 2018

1 Introduction

Energy efficiency is the ability to do more with less. When we talk about *green communications*, this means more connectivity, more capacity and more responsiveness at a lower energy cost. Unfortunately, these are competing goals, and all we can do in this tale is to fight against thermodynamics to seek for the elusive *optimality*.

It is straightforward to compute the minimum energy required to move a mass on a surface from point A to point B, as it depends on the friction coefficient. Drawing a parallelism, we are interested in the *friction*[1] of communicating a *bit* from point A to point B. As in the Newtonian example, our problem depends on the *substrate*, the specific technology and implementation. But when we deal with such complex systems as today's telecommunications networks, this estimation remains unfeasible.

We walk blindly towards an unknown lower bound. Fortunately, the set of possible strategies are limited and well-known[2], and pursue a common objective, namely, that the energy consumption should be proportional to the transmitted information. These can be divided in three basic principles subject of active research, and their intent can be summarised as follows:

Speed scaling, which tries to adapt the processing speed of network devices to the traffic load.

Powering down, which aims to turn network devices off during inactivity periods. This is commonly implemented as the so-called *sleep states*, and their *depth* introduces an inherent trade-off with the time required to become active again.

Rate adaptation, which tries to adapt the transmission rate of wireless cards according to the channel conditions[3].

The aim of this thesis is to explore the above three strategies in the scope of wireless communications for mobile user devices at multiple layers, as will be introduced in the following sections: from the operating system to the low-level operation of the wireless card.

[1] P. Grover. Information Friction and Its Implications on Minimum Energy Required for Communication. *IEEE Transactions on Information Theory*, 61(2): 895–907, Feb 2015. ISSN 0018-9448. DOI: 10.1109/TIT.2014.2365777

[2] K. Samdanis, P. Rost, A. Maeder, M. Meo, and C. Verikoukis. *Green Communications: Principles, Concepts and Practice*. Wiley & Sons, 2015. ISBN 9781118759240

[3] This, as well as the *speed scaling* principle, is based on a direct relationship between consumption and transmission rate, which is not always the case.

1.1 Cross-Factor: Towards a New Energy Model

We are living in an era in which consumer electronics are becoming *wireless consumer electronics*. That is, just about any known gadget is capable of connecting to cellular, wireless Local Area Networks (WLAN) or wireless Personal Area Networks (WPAN) nowadays. The Internet of Things (IoT) is growing fast and wireless communications become its main driver. Due to the densification of wireless networks and the ubiquity of battery-powered devices, energy efficiency stands as a major research issue.

More and more devices around us are becoming *smart*, incorporating more processing power in order to do more things, and some of them are already comparable to a small laptop computer. As a consequence, not only do they share hardware components, but also software: in particular, the Linux kernel is spreading into billions of devices all over the world, whether inside of the popular Android operating system or other embedded systems.

Whereas a lot of research were and is devoted to obtain more energy-efficient hardware components (e.g., wireless cards, processors, screens), too little attention has been paid, in terms of energy, to a core software component that enables all these devices: the operating system and, inside it, the network stack.

The seminal paper by Feeney and Nilsson [2001][4] gives the very first insight on energy consumption of wireless interfaces. This work was done on a per-frame basis by accounting for the energy drained by an 802.11 wireless card. Subsequent experimental work followed the same approach, showing that the energy consumption of wireless transmissions/receptions can be characterised using a simple linear model. This fact arises from the three typical states of operation of a wireless card: idle, transmitting and receiving.

The model, commonly expressed in terms of power, has a fixed part ρ_{id}, device-dependent, attributable to the idle state, and it grows linearly with the airtime percentage τ (i.e., the fraction of time in which the card is transmitting or receiving).

However, more recently, the novel work by Serrano et al. [2015][5] performed extensive per-frame measurements for seven devices of different types (smartphones, tablets, embedded devices and wireless routers), and unveiled that actually there is a non-negligible per-frame processing toll ascribed to the frame crossing the network stack. This component emerges as a new offset proportional to the frame generation rate, and it is not explained by the *classical* model.

[4] L. Feeney and 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*, volume 3, pages 1548–1557. IEEE, 2001. ISBN 0-7803-7016-3. DOI: 10.1109/INFCOM.2001.916651

[5] P. Serrano, A. Garcia-Saavedra, G. Bianchi, A. Banchs, and A. Azcorra. Per-Frame Energy Consumption in 802.11 Devices and Its Implication on Modeling and Design. *IEEE/ACM Transactions on Networking*, 23(4):1243–1256, Aug 2015. ISSN 1063-6692. DOI: 10.1109/TNET.2014.2322262

The currently accepted energy model is a multilinear model articulated into three main parts:

$$\overline{P}(\tau_i, \lambda_i) = \underbrace{\rho_{\text{id}} + \sum_{i \in \{\text{tx,rx}\}} \rho_i \tau_i}_{\text{classical model}} + \sum_{i \in \{\text{g,r}\}} \gamma_{\text{x}i} \lambda_i \tag{1.1}$$

where the first two addends correspond to the classical model and the third is the contribution described in Serrano et al. [2015]. This model can be subdivided into five components:

- A platform-specific baseline power consumption that accounts for the energy consumed just by the fact of being powered on, but with no network activity. This component is commonly referred to as *idle consumption*, ρ_{id}.
- A component that accounts for the energy consumed in transmission, which grows linearly with the airtime percentage τ_{tx}. The slope ρ_{tx} depends linearly on the radio transmission parameters MCS and TXP. $\overline{P_{\text{tx}}}(\tau_{\text{tx}}) = \rho_{\text{tx}} \tau_{\text{tx}}$
- A component that accounts for the energy consumed in reception, which grows linearly with the airtime percentage τ_{rx}. The slope ρ_{rx} depends linearly on the radio transmission parameter MCS. $\overline{P_{\text{rx}}}(\tau_{\text{rx}}) = \rho_{\text{rx}} \tau_{\text{rx}}$
- A new component, called *generation cross-factor* or γ_{xg}, that accounts for a per-frame energy processing toll in transmission, which grows linearly with the traffic rate λ_{g} generated. The slope γ_{xg} depends on the computing characteristics of the device. $\overline{P_{\text{xg}}}(\lambda_{\text{g}}) = \gamma_{\text{xg}} \lambda_{\text{g}}$
- A new component, called *reception cross-factor* or γ_{xr}, that accounts for a per-frame energy processing toll in reception, which grows linearly with the traffic rate λ_{r} received. Likewise, the slope γ_{xr} depends on the computing characteristics of the device. $\overline{P_{\text{xr}}}(\lambda_{\text{r}}) = \gamma_{\text{xr}} \lambda_{\text{r}}$

Therefore, the average power consumed $\overline{P}$ is a function of five device-dependent parameters ($\rho_i, \gamma_{\text{x}i}$) and four traffic-dependent ones (τ_i, λ_i).

The results obtained with this new multilinear model showed that the so-called cross-factor accounts for 50% to 97% of the per-frame energy consumption. Additional findings showed that it is independent of the CPU load on some devices and *almost* independent of the frame size.

Therefore, this inspired us to deepen into the roots of the cross-factor, to deseed its components and analyse its causes. To this aim, it is of paramount importance the conception of a comprehensive, high-accuracy and high-precision measurement framework that enables us to measure the consumption of a wide range of mobile devices.

1.2 *Micro-Sleep Opportunities in 802.11*

IEEE 802.11 is an extremely common technology for broadband Internet access. Energy efficiency stands as a major issue due to the intrinsic CSMA mechanism, which forces the network card to stay active performing *idle listening*, while most of the 802.11-capable terminals run on batteries.

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 11e amendment (Perez-Costa and Camps-Mur [2010][6] give 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 11ac amendment improves the PS capabilities with the VHT TXOP (Very High Throughput, Transmission Opportunity) PS mechanism. Basically, an 11ac STA can doze during other STAs' TXOPs. This capability is announced within the new VHT 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 waste[7]. Despite this, our measurements found no trace of mechanisms in commercial wireless card to lessen its impact.

Taking advantage on the energy measurement framework built for this thesis, we further investigate the timing capabilities for transitioning between operational states of a commercial off-the-shelf (COTS) wireless card. Based on this, we propose μNap, a practical algorithm to leverage micro-sleep opportunities in 802.11 WLANs.

[6] X. Perez-Costa and D. Camps-Mur. IEEE 802.11E QoS and power saving features overview and analysis of combined performance [Accepted from Open Call. *IEEE Wireless Communications*, 17(4):88–96, Aug. 2010. ISSN 1536-1284. DOI: 10.1109/MWC.2010.5547926

[7] P. Basu and 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*, IPSN '04, pages 196–204, New York, NY, USA, 2004. ACM. ISBN 1-58113-846-6. DOI: 10.1145/984622.984652

1.3 Rate Adaptation and Power Control in 802.11

At the beginning of this introduction, we have identified rate adaptation as one of the three basic strategies that can be applied to achieve proportionality between energy consumption and traffic load. Nevertheless, energy efficiency has not been the main driver for research in the fields of rate adaptation and power control in 802.11.

Rate adaptation (RA) algorithms are responsible for selecting the most appropriate modulation and coding scheme (MCS) to use given an estimation of the link conditions, being the frame losses one of the main indicators of such conditions. In general, the challenge lies in distinguishing between those losses due to collisions and those due to poor radio conditions, because they should trigger different reactions. In addition, the performance figure to optimise is commonly the throughput or a related one such as, e.g., the time required to deliver a frame.

On the other hand, network densification is becoming a common tool to provide better coverage and capacity. However, densification brings new problems, especially for 802.11, given the limited amount of orthogonal channels available, which leads to performance and reliability issues due to radio frequency interference. In consequence, some RA schemes also incorporate transmission power control (TPC), which tries to minimise the transmission power (TXP) with the purpose of reducing interference between nearby networks. As in the case of "vanilla" RA, the main performance figure to optimise is also the throughput.

It is generally assumed that optimality in terms of throughput also implies optimality in terms of energy efficiency. However, some previous work[8] has shown that throughput maximisation does not result in energy efficiency maximisation, at least for 802.11n. However, we still lack a proper understanding of the causes behind this "non-duality", as it may be caused by the specific design of the algorithms studied, an extra consumption caused by the complexity of MIMO techniques, or any other reason.

This thesis revisits RA-TPC in 802.11 within the context of energy efficiency. We first conduct an analytical study, and then we evaluate the performance, in terms of energy, of several state-of-the-art RA-TPC algorithms by simulation. Our results unveil that the trade-off shown by previous studies is in turn inherent to 802.11 operation. Particularly, we show that the link quality conditions which trigger mode transitions (changes of MCS/TXP) may be source of inefficiencies. Nonetheless, our analyses provide some heuristics that can be used along the energy parameters of the wireless card to mitigate such inefficiencies.

[8] C.-Y. Li, C. Peng, S. Lu, and X. Wang. Energy-based Rate Adaptation for 802.11n. In *Proceedings of the 18th Annual International Conference on Mobile Computing and Networking*, Mobicom '12, pages 341–352, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1159-5. DOI: 10.1145/2348543.2348585; and M. O. Khan, V. Dave, Y.-C. Chen, O. Jensen, L. Qiu, A. Bhartia, and S. Rallapalli. Model-driven energy-aware rate adaptation. In *Proceedings of the Fourteenth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, MobiHoc '13, pages 217–226, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2193-8. DOI: 10.1145/2491288.2491300

1.4 *Applied Simulation Modelling for Energy Efficiency*

Simulation frameworks are undoubtedly one of the most important tools for the analysis and design of communication networks and protocols. Their applications are numerous, including the performance evaluation of existing or novel proposals, dimensioning of resources and capacity planning, or the validation of theoretical analyses.

Simulation frameworks make a number of simplifying assumptions to reduce complexity so the development of the scenario is easier and numerical figures are obtained faster. This "complexity" axis goes from very specialised, large simulation tools such as NS-3, OMNeT++, OPNET, to *ad-hoc* simulation tools, consisting on hundreds of lines of code, typically used to validate a very specific part of the network or a given mathematical analysis. The latter are often developed over general-purpose languages such as C/C++ or Python, over numerical frameworks such as Matlab, or over some framework for discrete-event simulation (DES).

On the one hand, the complexity of specialised tools (as their cost, if applicable) preclude their use for short-to-medium research projects, as the learning curve is typically steep plus they are difficult to extend, which is mandatory to test a novel functionality. On the other hand, the development of ad-hoc tools also require some investment of time and resources, lack a proper validation of their functionality, and, furthermore, there is no code maintenance once the project is finished, for the few cases in which the code is made publicly available.

Last but not least, the monitoring of variables of interest typically stands out as a first-order problem. Most simulators require a specific effort in this regard, so that the modeller has to ponder which parameters must be monitored, and where and how this should be done. Therefore, the design of the scenario is intrinsically linked to data collection.

The research experience gained over the course of this thesis have made us aware of the need for new simulation tools committed to the middle-way approach: less specificity and complexity in exchange for faster prototyping, while supporting efficient simulation at a low implementation cost. As a result, we developed `simmer`, a process-oriented and trajectory-based DES package for R, which is designed to be an easy-to-use yet powerful framework. Its most noteworthy characteristic is the automatic monitoring capability, which allows the user to focus on the system model. The use of this simulator in networking is demonstrated through the energy modelling of a 5G-inspired scenario.

1.5 Thesis Overview

The remainder of this thesis is organised as follows. Chapter 2 presents the relevant literature related to the topics introduced in this chapter. Then, the dissertation is divided into three thematic parts: *experimentation*, *mathematical modelling* and *simulation*. Chapter 3 presents a comprehensive energy measurement framework, which has been a fundamental pillar of this thesis. Building on this, Chapter 4 delves into the roots of the cross-factor by exploring the energy consumption of the Linux network stack. Chapter 5 completes the experimental part with a study of the timing constraints of a COTS wireless card, which is developed into a standard-compliant algorithm to leverage micro-sleep opportunities in 802.11 WLANs. Chapter 6 presents a joint goodput-energy model that unveils an inherent trade-off between througput and energy efficiency maximisation in 802.11 when RA-TPC techniques are applied. Chapter 7 further extends the latter work by simulating and comparing the performance of several representative RA-TPC algorithms. Closing the simulation part, Chapter 8 presents our novel DES framework, and showcases its versatility by analising the energy consumption of an Internet-of-Things scenario with thousands of metering devices. Finally, Chapter 9 contains the thesis conclusions and future lines of work.

This thesis covers the following contributions:

Chapter 3

- **I. Ucar**, E. Pebesma, and A. Azcorra. Measurement Errors in R. *The R Journal*, 10(2):549–557, 2018c. ISSN 2073-4859. DOI: 10.32614/RJ-2018-075.

Chapter 4

- **I. Ucar**, A. Azcorra, and A. Banchs. Deseeding Energy Consumption of Network Stacks. In *6th Annual International IMDEA Networks Workshop*, June 2014. (invited poster).
- **I. Ucar** and A. Azcorra. Deseeding energy consumption of network stacks. In *IEEE 1st International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI)*, pages 7–16, Sept. 2015. ISBN 978-1-4673-8167-3. DOI: 10.1109/RTSI.2015.7325085.

Chapter 5

- A. Azcorra, **I. Ucar**, A. Banchs, F. Gringoli, and P. Serrano. μNap: Practical micro-sleeps for 802.11 WLANs. *Computer Communications*, 110:175–186, Sept. 2017. ISSN 0140-3664. DOI: 10.1016/j.comcom.2017.06.008.
- A. Azcorra, **I. Ucar**, A. Banchs, F. Gringoli, and P. Serrano. Energy-saving method based on micro-shutdowns for a wireless device in a telecommunications network. WO/2018/015601, Jan. 2018. URL `http://www.google.com/patents/WO2018015601`.

Chapter 6

- **I. Ucar**, C. Donato, P. Serrano, A. Garcia-Saavedra, A. Azcorra, and A. Banchs. Revisiting 802.11 Rate Adaptation from Energy Consumption's Perspective. In *Proceedings of the 19th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, MSWiM '16, pages 27–34, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4502-6. DOI: 10.1145/2988287.2989149.

Chapter 7

- **I. Ucar**, C. Donato, P. Serrano, A. Garcia-Saavedra, A. Azcorra, and A. Banchs. On the energy efficiency of rate and transmission power control in 802.11. *Computer Communications*, 117:164–174, Feb. 2018a. ISSN 0140-3664. DOI: 10.1016/j.comcom.2017.07.002.

Chapter 8

- **I. Ucar**, B. Smeets, and A. Azcorra. simmer: Discrete-Event Simulation for R. *Journal of Statistical Software*, (accepted for publication), 2018d. ISSN 1548-7660. URL `https://arxiv.org/abs/1705.09746`.
- **I. Ucar**, J. A. Hernández, P. Serrano, and A. Azcorra. Design and Analysis of 5G Scenarios with simmer: An R Package for Fast DES Prototyping. *IEEE Communications Magazine*, 56(11):145–151, Nov. 2018b. ISSN 0163-6804. DOI: 10.1109/MCOM.2018.1700960.

Therefore, there is an overlap between this dissertation and the publications listed above. Additionally, the following contributions are not part of this thesis:

- N. Bartzoudis, O. Font-Bach, M. Miozzo, C. Donato, P. Harbanau, M. Requena, D. López, **I. Ucar**, A. Azcorra, P. Serrano, J. Mangues, and M. Payaró. Energy footprint reduction in 5G reconfigurable hotspots via function partitioning and bandwidth adaptation. In *2017 Fifth International Workshop on Cloud Technologies and Energy Efficiency in Mobile Communication Networks (CLEEN)*, pages 1–6, June 2017. DOI: 10.23919/CLEEN.2017.8045934.
- C. Mannweiler, M. Breitbach, H. Droste, I. L. Pavón, **I. Ucar**, P. Schneider, M. Doll, and J. R. Sanchez. 5G NORMA: System architecture for programmable multi-tenant 5G mobile networks. In *2017 European Conference on Networks and Communications (EuCNC)*, pages 1–6, June 2017. DOI: 10.1109/EuCNC.2017.7980662.
- F. Gringoli, P. Serrano, **I. Ucar**, N. Facchi, and A. Azcorra. Experimental QoE evaluation of multicast video delivery over IEEE 802.11aa WLANs. *IEEE Transactions on Mobile Computing*, Early Access, 2018. ISSN 1536-1233. DOI: 10.1109/TMC.2018.2876000.

2 Related Work

In recent years, we have witnessed an increased attention towards "green operation" of networks, which is required to support a sustainable growth of the communication infrastructures. For the case of wireless communications, there is the added motivation of a limited energy supply (i.e., batteries) in user mobile devices, which has triggered a relatively large amount of work on energy efficiency[1].

[1] P. Serrano, A. de la Oliva, P. Patras, V. Mancuso, and A. Banchs. Greening wireless communications: Status and future directions. *Computer Communications*, 35(14):1651 – 1661, 2012. ISSN 0140-3664. DOI: 10.1016/j.comcom.2012.06.011. Special issue: Wireless Green Communications and Networking

2.1 Energy Profiling for Wireless Communications

Energy profiling is the most fundamental work on green communications. In a similar way as *software profiling* measures the duration of function calls, *energy profiling* measures the energy consumed by each software or hardware *function*, in the broad sense of the word.

There are two energy measurement techniques[2]: software- and hardware-assisted measurements. The former uses the built-in battery interface integrated in many modern devices (e.g., smartphones, tablets), which provides readings of the dicharge curve. Apart from the evident feedback loop problem[3], this method lacks granularity and precision, and therefore it is not suited for fine-grained energy profiling. Despite this, it has been used in a number of papers, such as Nurminen and Noyranen [2008], Xiao et al. [2008], Balasubramanian et al. [2009] and Kalic et al. [2012], some of them based on Creus and Kuulusa [2007]. Dong and Zhong [2011], Jung et al. [2012] and Xu et al. [2013] try to overcome this limitations by developing an online power model on top of the battery monitoring unit.

[2] S. Tarkoma, M. Siekkinen, E. Lagerspetz, and Y. Xiao. *Smartphone Energy Consumption: Modeling and Optimization*. Smartphone Energy Consumption: Modeling and Optimization. Cambridge University Press, 2014. ISBN 9781107042339

[3] Namely, that the software used to collect such readings is in fact producing an energy overhead.

Instead, in hardware-assisted measurements, the battery is bypassed to directly measure the current and voltage[4] signals. The current signal is converted to voltage with a small high-precision resistor, and then sampled with a variety of instruments (digital multimeters, oscilloscopes, data acquisition devices...). Thus, the granularity and precision is limited by the specific setup and instrumentation selected.

[4] User mobile devices use DC power. Therefore, the voltage signal is usually ignored and assumed to be constant, which may not be the best approach for fine-grained measurements.

Typically, energy profiling studies in wireless communications provide a measurement setup that is expressly designed for the electrical specifications of the device under test. For instance, Feeney and Nilsson [2001] and Ebert et al. [2002] use a cardbus extender and an oscilloscope to measure wireless interfaces. Carroll and Heiser [2010], Rice and Hay [2010], Wang and Manner [2010] perform energy measurements on smartphones using a data acquisition (DAQ) card, while Gupta and Mohapatra [2007] used a power meter.

More recent studies (e.g., Zhang et al. [2010], Pathak et al. [2011b], Manweiler and Roy Choudhury [2011], Jung et al. [2012], Oliner et al. [2012], Xu et al. [2013], Kim et al. [2013], Zhang et al. [2013]) started using the Monsoon[5] Power Monitor, a product specifically designed to test smartphones which has become very popular. There have also been initiatives to design open-source energy measurement and control systems such as Energino[6] [Riggio et al., 2012, Gomez et al., 2012].

[5] https://www.msoon.com/

[6] http://www.energino-project.org/

2.2 *Energy Consumption of Network Stacks*

The *classical* energy model for wireless interfaces was first established by Feeney and Nilsson [2001], and further confirmed by Ebert et al. [2002] and Shih et al. [2002]. Since then, it has been assumed that the network card dominates the consumption of wireless communications of mobile user devices.

This model has been widely used (implicitly or explicitly) in tens of papers to justify energy savings with optimisations of diverse kind: PHY layer rate and power [Qiao et al., 2003], MAC parameters [Bruno et al., 2002, Carvalho et al., 2004, Agrawal, 2004, Jyh-Cheng Chen and Kai-wen Cheng, 2008, Garcia-Saavedra et al., 2011], backoff operation [Vaidya, 2002, Baiamonte and Chiasserini, 2006], idle state [Zhang and Shin, 2012], packet overhearing [Ergen and Varaiya, 2007], packet relying [He and Li, 2010], data compression [Baek et al., 2004, Sharma et al., 2009], etcetera.

However, the work by Serrano et al. [2015], which provides the *new* energy model studied in the previous chapter, poses doubts on prior work that propose energy efficiency strategies taking into account the wireless card only. Bearing in mind these results, it is imperative to reconsider existing schemes for energy efficiency in wireless communications. Old schemes like packet relying [He and Li, 2010] and data compression [Baek et al., 2004, Sharma et al., 2009] may no longer be valid under the new model. On the contrary, packet batching or low-level packet generation could potentially produce real savings.

2.3 *Micro-Sleep Opportunities in 802.11*

It is well-known 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 [2008] 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. [2011] 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 [2012] 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[7].

[7] See Section 3.4.1.

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 [Kamerman and Monteban, 1997a, Havinga and Smit, 2000, Jung and Vaidya, 2002].

Baiamonte and Chiasserini [2006] 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 back-off 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. [2010] 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, Prasad et al. [2014] proposed Über-sleep. 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 et al. [2013b] and Palacios et al. [2013a] modified 802.11 DCF and PCF to exploit per-packet sleeps. They also applied these ideas to network coding [Palacios-Trujillo et al., 2015] and to a polling-based version of 11ac's TXOP PS mode [Palacios et al., 2015]. Unfortunately, all these papers rely on these early studies mentioned before [Kamerman and Monteban, 1997a, Havinga and Smit, 2000, Jung and Vaidya, 2002], which analysed old wireless cards unable to perform sub-millisecond transitions between states.

2.4 Rate Adaptation and Power Control in 802.11

Rate adaptation and power control in 802.11 have received a vast amount of attention from the research community (especially the former). See, e.g., Biaz and Wu [2008], Huang et al. [2013] and references therein. But, as stated before, all this research is generally focused towards performance (i.e., throughput and capacity) maximisation.

However, as some previous work has pointed out [Eryigit et al., 2014, Garcia-Saavedra et al., 2012], energy efficiency and performance do not necessarily come hand in hand, and some criterion may be required to set a proper balance between them. Particularly, Li et al. [2012] and Khan et al. [2013] studied RA for energy efficiency in 802.11n, and showed that MIMO RA algorithms are not energy efficient despite ensuring high throughput. These algorithms incur inefficiencies when marginal throughput gains are achieved at a high energy cost.

No other studies have further investigated these inefficiencies in more detail though. The question remains, then, whether there may be fundamental trade-offs in the application of RA-TPC techniques to 802.11.

2.5 Discrete-Event Simulation of Network Systems

The list of network simulation tools is vast[8]. Particularly, OPNET[9] is one of the most complex and widely used tools for research and education, although the open-source alternatives, such as NS3 and OMNeT++, are growing and becoming more prominent within the research community. As an illustrative comparison, a quick search on Google Scholar[10] returns over 30k results for "opnet simulation", about 6k for "omnet++ simulation" and about 12k for "ns3 simulation". Not surprisingly, the old version of the latter, NS2, still produces more than 40k results. Still, the amount of research produced by the networking community is enormous, and it is difficult to quantify how much of that research is done using ad-hoc simulation tools.

Besides that, the list of general-purpose DES frameworks is even more vast[11], and most of them are unknown within the networking community because they are generally more suited for other fields, such as operations research. Among them, the `SimPy` Python package[12] has received increased attention, and has inspired similar frameworks for other languages, such as `SimJulia` [Lauwens, 2017]. Nonetheless, some studies showed that SimPy may not scale for large network simulations [Bahouth et al., 2007, Weingartner et al. [2009]].

[8] See, e.g., `http://people.idsia.ch/~andrea/sim/simnet.html`.

[9] `http://www.opnet.com`

[10] As of April 2018.

[11] See, e.g., `https://en.wikipedia.org/wiki/List_of_discrete_event_simulation_software`.

[12] Team SimPy. *SimPy: Discrete-Event Simulation for Python*, 2017. URL `https://simpy.readthedocs.io/en/latest/`. Python package version 3.0.9

Part I Experimentation

3 A Comprehensive Energy Measurement Framework

Energy measurements are typically conducted in an *ad-hoc* manner, with hardware and software tools specifically designed for a particular use case, and thus for limited electrical specifications. In this thesis, we are interested in measuring from small hardware components, such as wireless interfaces, to *energy debugging*[1], which implies whole-device mesurements. At the same time, potential wireless devices range from wearables and smartphones to access points and laptop computers. As a result, a flexible and reusable energy measurement framework for wireless communications must be able to cover a wide spectrum of electrical specifications (current, voltage and power) without losing accuracy or precision.

The main specifications for our framework are the following:

- High-accuracy, high-precision; in the range of mW.
- Avoid losing information between sampling periods, with events (transmission and reception of wireless frames) that last tens of microseconds.
- Support for a wide range of devices, from low- to high-powered devices, while keeping accuracy and precision.
- Support for synchronous measurements of multiple devices under test (DUTs), for network-related energy measurements (e.g., protocol/algorithm testing, energy optimisation of a network as a whole) as well as *stacked* measurements (e.g., measuring different components of a device at the same time).

Based on the above, we fist describe the selected instrumentation. A brief discussion follows about how to handle measurements and their associated uncertainty. We develop a method for automatic propagation and representation of uncertainty within the R language[2]. Then, a testbed for whole-device measurements is proposed and validated. Finally, a complementary setup for per-component measurements is described and demonstrated by characterising a wireless interface.

[1] A. Pathak, Y. C. Hu, and M. Zhang. Bootstrapping energy debugging on smartphones. In *Proceedings of the 10th ACM Workshop on Hot Topics in Networks - HotNets '11*, pages 1–6, New York, New York, USA, Nov. 2011a. ACM Press. ISBN 9781450310598. DOI: 10.1145/2070562.2070567

[2] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2017. URL https://www.R-project.org/

3.1 *Instrumentation*

Although there are integrated solutions in the market, known as *power analysers*, these are expensive instruments mainly aimed at characterising AC power (e.g., power factor, harmonics). Instead, we are interested in (time varying) DC power, and breaking the problem into its constituent parts (i.e., power supply, signal adaptation and data acquisition) enables not only more flexibility, but a wider choice at lower prices.

The most power-hungry devices within the scope of this thesis are laptop computers. Most of these devices rarely surpass the barrier of 100 W[3]. Therefore, we selected the Keithley 2304A DC Power Supply, which is optimised for testing battery-operated wireless communication devices[4] that undergo substantial load changes for very short time intervals. This power supply simulates a battery's response during a large load change by minimising the maximum drop in voltage and recovering to within 100 mV of the original voltage within 40 μs.

As the measurement device, we selected the National Instruments PCI-6289 card, a high-accuracy multifunction data acquisition (DAQ) device. It has 32 analogue inputs (16 differential or 32 single ended) with 7 input ranges optimised for 18-bit input accuracy, up to 625 kS/s single channel or 500 kS/s multi-channel (aggregate). The timing resolution is 50 ns with an accuracy of 50 ppm of sample rate.

Finally, the required signals (current and voltage) must be extracted and adapted to the DAQ's input specifications. A custom three-port circuit, specifically designed by our university's Technical Office in Electronics, converts the current signal to voltage, and adapts the voltage signal to the DAQ's input limits without loss of precision. Two distinct designs, with different input specifications, were built (see Appendix A for further details).

Figure 3.1 shows a simplified scheme of this circuit. The voltage drop in a small and high-precision resistor is amplified to measure the current signal. At the same time, a resistive divider couples the voltage signal. Considering that the DAQ card has certain settling time, it can be modelled as a small capacity 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 circuit[5].

A small command-line tool[6] was developed to perform measurements on the DAQ card using the open-source Comedi[7] drivers and libraries.

Regarding the kernel instrumentation, we take advantage of SystemTap[8], an open-source infrastructure around the Linux kernel that dramatically simplifies information gathering on a running Linux system (kernel, modules and applications). It provides a scripting

[3] For instance, typical Dell computers bring 65 or 90 W AC adapters with maximum voltages of 19.5 V.

[4] Up to 100 W, 20 V, 5 A.

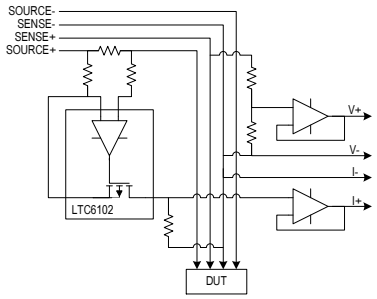


Figure 3.1: Measurement circuit (simplified) devoted to extract and adapt the signals to the DAQ input requirements.

[5] National Instruments. Using a Unity Gain Buffer (Voltage Follower) with a DAQ Device. White paper, Jan. 2014. URL http://www.ni.com/white-paper/4494/en/

[6] Available at https://github.com/Enchufa2/daq-acquire

[7] http://comedi.org

[8] https://sourceware.org/systemtap

language for writing instrumentation. A SystemTap script is parsed into C code, compiled into a kernel module and hot-plugged into a live running system.

3.2 *Measurement and Uncertainty Analysis*

The International Vocabulary of Metrology (VIM) defines a *quantity* as follows[9]:

> *A property of a phenomenon, body, or substance, where the property has a magnitude that can be expressed as a number and a reference.*

where most typically the number is a *quantity value*, attributed to a *measurand* and experimentally obtained via some measurement procedure, and the reference is a *measurement unit*.

Additionally, any quantity value must accommodate some indication about the quality of the measurement, a quantifiable attribute known as *uncertainty* (also traditionally known as *error*). The Guide to the Expression of Uncertainty in Measurement (GUM) defines *uncertainty* as follows[10]:

> *A parameter, associated with the result of a measurement, that characterises the dispersion of the values that could reasonably be attributed to the measurand.*

Uncertainty can be mainly classified into *standard uncertainty*, which is the result of a direct measurement (e.g., electrical voltage measured with a voltmeter, or current measured with a amperimeter), and *combined standard uncertainty*, which is the result of an indirect measurement (i.e., the standard uncertainty when the result is derived from a number of other quantities by the means of some mathematical relationship; e.g., electrical power as a product of voltage and current). Therefore, provided a set of quantities with known uncertainties, the process of obtaining the uncertainty of a derived measurement is called *propagation of uncertainty*.

Traditionally, computational systems have treated these three components (quantity values, measurement units and uncertainty) separately. Data consisted of bare numbers, and mathematical operations applied to them solely. Units were just metadata, and error propagation was an unpleasant task requiring additional effort and complex operations. Nowadays though, many software libraries have formalised *quantity calculus* as method of including units within the scope of mathematical operations, thus preserving dimensional correctness and protecting us from computing nonsensical combinations of quantities. However, these libraries rarely integrate uncertainty handling and propagation[11].

[9] BIPM, IEC, IFCC, ILAC, IUPAC, IUPAP, ISO, and OIML. The International Vocabulary of Metrology – Basic and General Concepts and Associated Terms (VIM), 3rd edn. JCGM 200:2012. *Joint Committee for Guides in Metrology*, 2012. URL http://www.bipm.org/vim

[10] BIPM, IEC, IFCC, ILAC, IUPAC, IUPAP, ISO, and OIML. Evaluation of Measurement Data – Guide to the Expression of Uncertainty in Measurement, 1st edn. JCGM 100:2008. *Joint Committee for Guides in Metrology*, 2008. URL https://www.bipm.org/en/publications/guides/gum.html

[11] D. Flater. Architecture for software-assisted quantity calculus. *Computer Standards & Interfaces*, 56:144–147, 2018. ISSN 0920-5489. DOI: 10.1016/j.csi.2017.10.002

Within the R environment, the `units` package[12] defines a class for associating unit metadata to numeric vectors, which enables transparent quantity derivation, simplification and conversion. This approach is a very comfortable way of managing units with the added advantage of eliminating an entire class of potential programming mistakes. Unfortunately, neither `units` nor any other package address the integration of uncertainties into quantity calculus.

In the following, we discuss propagation and reporting of uncertainty, and we present a framework for associating uncertainty metadata to R vectors, matrices and arrays, thus providing transparent, lightweight and automated propagation of uncertainty. This implementation also enables ongoing developments for integrating units and uncertainty handling into a complete solution.

[12] E. Pebesma and T. Mailund. *units: Measurement Units for R Vectors*, 2018. URL https://CRAN.R-project.org/package=units. R package version 0.5-1; and E. Pebesma, T. Mailund, and J. Hiebert. Measurement Units in R. *The R Journal*, 8(2):486–494, 2016. URL https://journal.r-project.org/archive/2016/RJ-2016-061/index.html

3.2.1 *Propagation of Uncertainty*

There are two main methods for propagation of uncertainty: the *Taylor series method* (TSM) and the *Monte Carlo method* (MCM). The TSM, also called the *delta method*, is based on a Taylor expansion of the mathematical expression that produces the output variables. As for the MCM, it is able to deal with generalised input distributions and propagates the error by Monte Carlo simulation.

The TSM is a flexible method of propagation of uncertainty that can offer different degrees of approximation given different sets of assumptions. The most common and well-known form of TSM is a first-order TSM assuming normality, linearity and independence. In the following, we will provide a short description. A full derivation, discussion and examples can be found in Arras [1998].

Mathematically, an indirect measurement is obtained as a function of n direct or indirect measurements, $Y = f(X_1, ..., X_n)$, where the distribution of X_n is unknown *a priori*. Usually, the sources of random variability are many, independent and probably unknown as well. Thus, the central limit theorem establishes that an addition of a sufficiently large number of random variables tends to a normal distribution. As a result, the *first assumption* states that X_n are normally distributed.

The *second assumption* presumes linearity, i.e., that f can be approximated by a first-order Taylor series expansion around μ_{X_n} (see Figure 3.2). Then, given a set of n input variables X and a set of m output variables Y, the first-order *error propagation law* establishes that

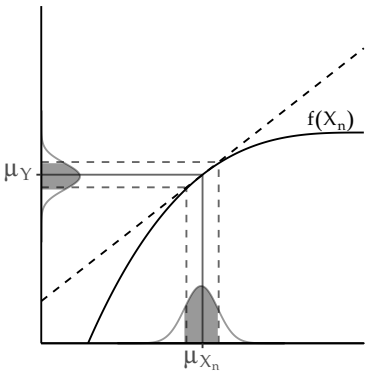


Figure 3.2: Illustration of linearity in an interval ± one standard deviation around the mean.

$$\Sigma_Y = J_X \Sigma_X J_X^T \tag{3.1}$$

where Σ is the covariance matrix and J is the Jacobian operator.

Finally, the *third assumption* supposes independency among the uncertainty of the input variables. This means that the cross-covariances are considered to be zero, and the equation above can be simplified into the most well-known form of the first-order TSM:

$$(\Delta y)^2 = \sum_i \left(\frac{\partial f}{\partial x_i} \right)^2 \cdot (\Delta x_i)^2 \tag{3.2}$$

In practice, as recommended in the GUM, this first-order approximation is good even if f is non-linear, provided that the non-linearity is negligible compared to the magnitude of the uncertainty, i.e., $\mathbb{E}[f(X)] \approx f(\mathbb{E}[X])$. Also, this weaker condition is distribution-free: no assumptions are needed on the probability density functions (PDF) of X_n, although they must be reasonably symmetric.

3.2.2 *Reporting Uncertainty*

The GUM defines four ways of reporting standard uncertainty and combined standard uncertainty. For instance, if the reported quantity is assumed to be a mass m_S of nominal value 100 g:

1. $m_S = 100.02147$ g with (a combined standard uncertainty) $u_c = 0.35$ mg.
2. $m_S = 100.02147(35)$ g, where the number in parentheses is the numerical value of (the combined standard uncertainty) u_c referred to the corresponding last digits of the quoted result.
3. $m_S = 100.02147(0.00035)$ g, where the number in parentheses is the numerical value of (the combined standard uncertainty) u_c expressed in the unit of the quoted result.
4. $m_S = (100.02147 \pm 0.00035)$ g, where the number following the symbol $\pm$ is the numerical value of (the combined standard uncertainty) u_c and not a confidence interval.

Schemes (2, 3) and (4) will be referred to as *parenthesis* notation and *plus-minus* notation respectively. Although (4) is a very extended notation, the GUM explicitly discourages its use to prevent confusion with confidence intervals. Throughout this document, we will be using (2) unless otherwise specified.

3.2.3 *Automated Uncertainty Handling in R: The `errors` Package*

Following the approach of the `units` package, this thesis develops a framework for automatic propagation and reporting of uncertainty: the `errors` package[13]. This R package aims to provide easy and lightweight handling of measurements with errors, including propagation using the first-order TSM presented in the previous section and a formally sound representation. Errors, given as (combined) standard uncertainties, can be assigned to numeric vectors, matrices

[13] **I. Ucar**. *errors: Uncertainty Propagation for R Vectors*, 2018. URL https://CRAN.R-project.org/package=errors. R package version 0.2.1

and arrays, and then all the mathematical and arithmetic operations are transparently applied to both the values and the associated errors. The following example sets a simple vector with a 5% of error:

```
library(errors)

x <- 1:5
errors(x) <- x * 0.05
x
```

```
## Errors: 0.05 0.10 0.15 0.20 0.25
## [1] 1 2 3 4 5
```

The `errors()` function assigns or retrieves a vector of errors, which is stored as an attribute of the class `errors`. Internally, the package provides S3 methods[14] for the generics belonging to the groups `Math`, `Ops` and `Summary`, plus additional operations such as subsetting (`[`, `[<-`, `[[`, `[[<-`), concatenation (`c()`), differentiation (`diff`), row and column binding (`rbind`, `cbind`), or coercion to data frame and matrix.

[14] See "Writing R Extensions" for further details: https://cran.r-project.org/doc/manuals/r-release/R-exts.html

```
data.frame(x, 3*x, x^2, sin(x), cumsum(x))
```

```
##         x  X3...x    x.2   sin.x. cumsum.x.
## 1 1.00(5)  3.0(2) 1.0(1)  0.84(3)   1.00(5)
## 2  2.0(1)  6.0(3) 4.0(4)  0.91(4)    3.0(1)
## 3  3.0(2)  9.0(5) 9.0(9)   0.1(1)    6.0(2)
## 4  4.0(2) 12.0(6)  16(2)  -0.8(1)   10.0(3)
## 5  5.0(2) 15.0(8)  25(2) -0.96(7)   15.0(4)
```

It is worth noting that both values and errors are stored with all the digits. However, when a single measurement or a column of measurements in a data frame are printed, the output is properly formatted to show a single significant digit for the error. This is achieved by providing S3 methods for `format()` and `print()`. The *parenthesis* notation is used by default, but this can be overridden through the appropriate option in order to use the *plus-minus* notation instead.

3.3 Whole-Device Measurements

Figure 3.3 shows the proposed testbed for whole-device measurements. It comprises two laptop computers —the DUT and an access point (AP)— and a controller. The controller is a workstation with the DAQ card installed and it performs the energy measurements. At the same time, it sends commands to the DUT and AP through a wired connection and monitors the wireless connection between DUT and AP through a probe.

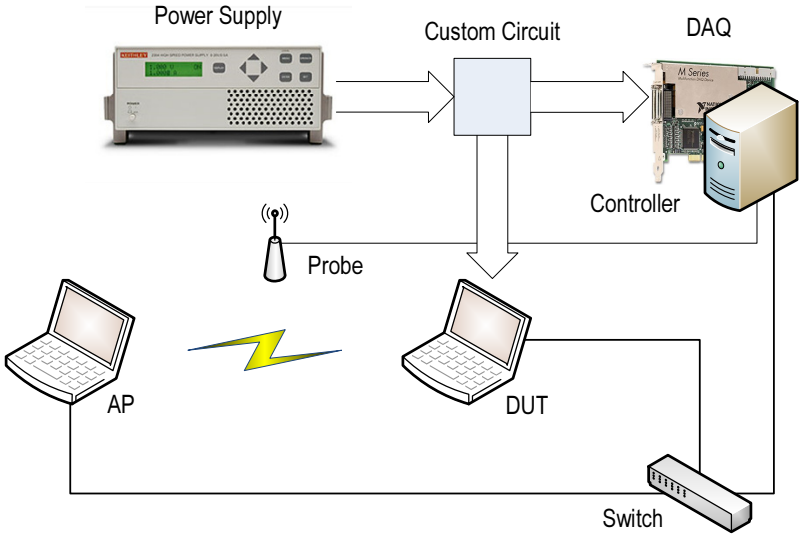


Figure 3.3: Testbed for whole-device energy measurements. The *custom circuit* is the one sketched in Figure 3.1.

The experimental methodology to characterise the DUT's energy parameters is as follows. Given a collection of network parameter values (modulation coding scheme or MCS, transmission power, packet size, framerate), we run steady experiments for several seconds in order to gather averaged measures. Each experiment comprises the steps shown in Figure 3.4.

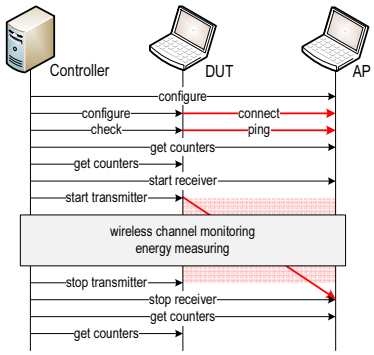


Figure 3.4: Measurement methodology. Time sequence of a whole-device experiment.

1. AP and DUT are configured. The DUT connects to the wireless network created by the AP and checks the connectivity. Setting up this network in a clear channel is highly advisable to avoid interference. The 5 GHz band, with an 802.11a-capable card, has good candidates.
2. The packet counters of the wireless interfaces are saved for later use.
3. Receiver and transmitter are started. We use the `mgen`[15] traffic generator and a simple `netcat` at the receiver.
4. The controller monitors the wireless channel and collects an energy trace that will be averaged later.
5. Transmitter and Receiver are stopped.
6. Because of the unreliability of the wireless medium, the packet counters, together with the monitoring information, are used to ensure that the experiment was successful (i.e., the traffic seen agrees with the configured parameters).

[15] `http://cs.itd.nrl.navy.mil/work/mgen`

3.3.1 *Validation*

In order to validate our measurement framework, several experiments were performed with one of the devices studied in Serrano

et al. [2015] as DUT. We selected the Soekris net4826-48 equipped with an Atheros AR5414-based 802.11a/b/g Mini-PCI card because it is the one with the largest cross-factor. The operating system (OS) was Linux Voyage with kernel 2.6.30 and the MadWifi driver v0.9.4.

The first task was to perform the energy breakdown given in Serrano et al. [2015] in transmission mode:

User space The Soekris generates packets using `mgen`, but they are discarded before being delivered to the OS, by using the `sink` device rather than `udp`.

Kernel space Packets cross the network stack and are discarded in the driver, by commenting the `hardstart` MadWifi command that performs the actual delivery of the frame to the wireless network interface card (NIC).

Wireless NIC Packets are transmitted, i.e., are delivered to the wireless medium.

The NoACK functionality from 802.11e was activated in order to avoid ACK receptions. Therefore, Equation (1.1) can be simplified as follows to describe complete transmissions:

$$\overline{P}(\tau, \lambda) = \rho_{id} + \rho_{tx}\tau + \gamma_{xg}\lambda$$

Figure 3.5 represents the equation above (red lines) and depicts how the energy toll splits across the processing chain with different parameters (blue and green lines). The dashed line depicts the idle consumption as a reference, $\rho_{id} = 3.65(1)$ W.

Indeed, these results are quite similar to Serrano et al. [2015] and confirm that the cross-factor accounts for the largest part of the energy consumption. Moreover, Serrano et al. [2015] reports that the cross-factor is *almost* independent of the packet size. Interestingly,

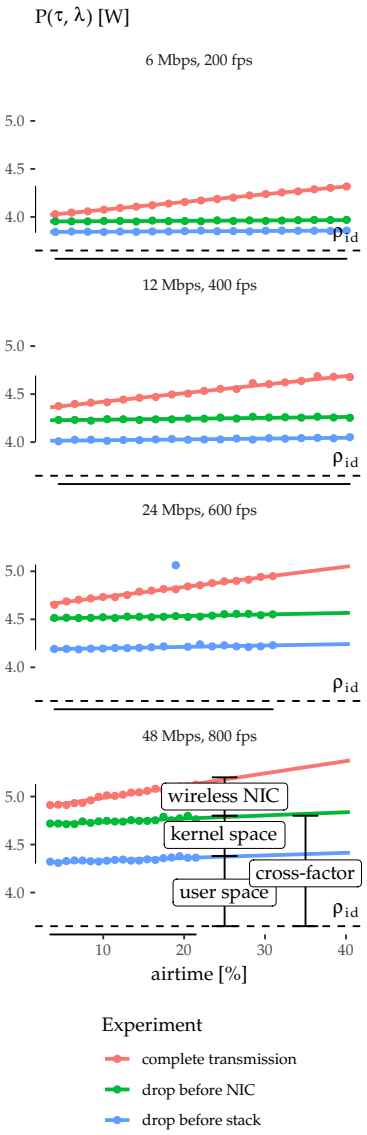


Figure 3.5: Power consumption breakdown vs. airtime.

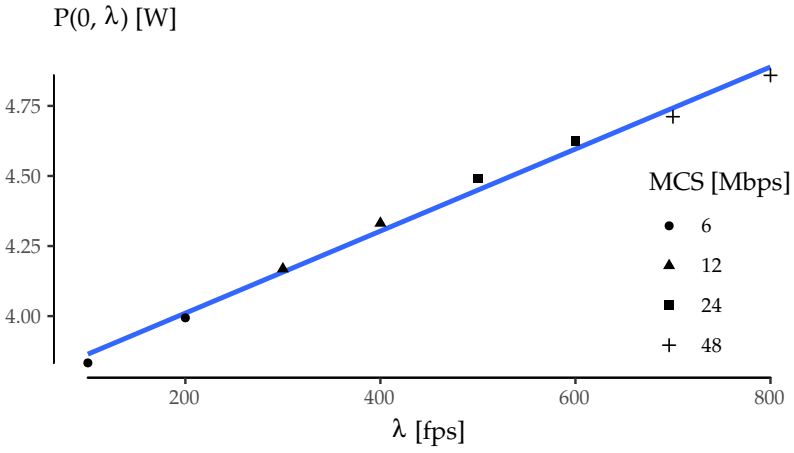


Figure 3.6: Power consumption offset ($\tau = 0$) vs. framerate.

our results have captured a small dependence that can be especially observed in the 600 fps case.

Finally, we can derive the cross-factor value and compare it. Taking the offset of the red regression lines of Figure 3.5, we can plot Figure 3.6 and fit these points with $\tau = 0$. This regression yields the values $\rho_{id} = 3.72(4)$ W (3.65(1) W measured) and $\gamma_{xg} = 1.46(7)$ mJ, quite close to the values reported in Serrano et al. [2015].

3.4 *Per-Component Measurements*

Figure 3.7 shows the proposed testbed for per-component measurements. The component (a wireless card) is attached to the device through a flexible x1 PCI Express to Mini PCI Express adapter from Amfeltec. 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.

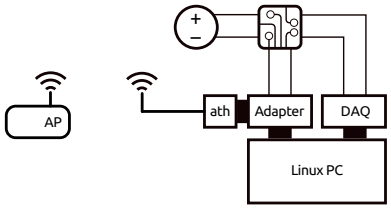


Figure 3.7: Testbed for per-component energy measurements.

Here, the same PC holds the DAQ card. In this way, the operations sent to the wireless card and the energy measurements can be correlated using the same timebase, which is required for the next section. For other type of experiments, this requirement can be relaxed, and the DAQ card can be hosted in a separate machine.

3.4.1 *Characterisation of a COTS Device*

State Consumption Parametrisation In the following, we demonstrate a complete state parametrisation (power consumption in transmission, reception, overhearing, idle and sleep) of a commercial off-the-shelf (COTS) card: an Atheros AR928o-based 802.11a/b/g/n Half Mini-PCI Express card. 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/transission 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 average power consumption was measured with a basic precision of 1 mW over intervals of 3 s.

Regarding the sleep state, the card's `ath9k` driver 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 Power Save (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 the traffic indication map), as required by the PS mode.

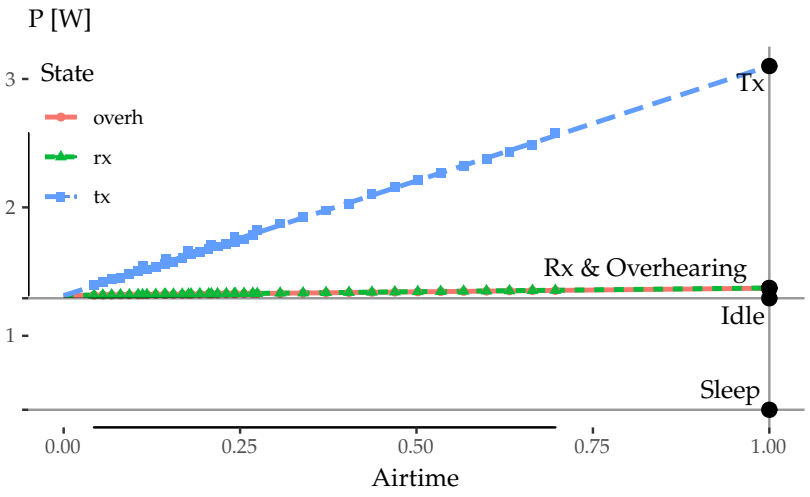


Figure 3.8: Atheros AR9280 power consumption in 11a mode.

Figure 3.8 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 average values are shown in Table 3.1. 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.

Downclocking Consumption Characterisation As the AR9280's documentation states, its reference clock runs at 44 MHz for 20 MHz

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

Table 3.1: Atheros AR9280 power consumption.

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 3.1 shows, we measured two more results to gain additional insight into the behaviour of the main reference clock, which is known to be linear[16].

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 work).

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:

$$P(f) = 0.91(3) + 0.0051(5)f \tag{3.3}$$

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 the one developed by Zhang and Shin [2012], will not save too much energy compared to the sleep state of COTS devices. For instance, the x16 downclocking achieved by this mechanism 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 3.1, 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.

[16] X. Zhang and K. G. Shin. E-MiLi: Energy-Minimizing Idle Listening in Wireless Networks. *IEEE Transactions on Mobile Computing*, 11(9):1441–1454, Sept. 2012. ISSN 1536-1233. DOI: 10.1109/TMC.2012.112

3.5 *Summary*

We have built and validated a comprehensive, high-accuracy and high-precision energy measurement framework, which is capable of measuring a wide range of wireless devices, as well as multiple heterogeneous devices synchronously. Rigorous experimental methodologies have been introduced to characterise the energy parameters of devices and wireless components. Based on these, the framework has been validated against previous results from Serrano et al. [2015], and a COTS wireless card, which will be used in further experiments throughout this thesis, has been studied and parametrised.

At the same time, measurement handling has been systematised into `errors`[17], a lightweight R package for managing numeric data with associated standard uncertainties. The new class `errors` provides numeric operations with automated propagation of uncertainty through a first-order TSM, and a formally sound representation of measurements with errors. Using this package makes the process of computing indirect measurements easier and less error-prone.

Future work includes importing and exporting data with uncertainties, and providing the user with an interface for plugging uncertainty propagation methods from other packages. Finally, `errors` enables ongoing developments[18] for integrating `units` and uncertainty handling into a complete solution for quantity calculus. Having a unified workflow for managing measurements with units and errors would be an interesting addition to the R ecosystem with very few precedents in other programming languages.

[17] **I. Ucar**, E. Pebesma, and A. Azcorra. Measurement Errors in R. *The R Journal*, 10(2):549–557, 2018c. ISSN 2073-4859. DOI: 10.32614/RJ-2018-075

[18] *Quantities for R*, a project funded by the R Consortium (see `https://www.r-consortium.org/projects/awarded-projects`)

4 Deseeding Energy Consumption of Network Stacks

The cross-factor is a per-frame energy toll that can be ascribed to the fact that every frame received or transmitted through a wireless interface requires some processing in the device's network stack. The aim of this chapter is to break down the cross-factor into its constituent parts in order to comprehend the underlying causes, all within the scope of providing an accurate mathematical description of the cross-factor which would give the energy model an unprecedented specificity. This may enable us to evaluate old energy efficiency strategies and to propose and test new schemes, both at a device and at a network level.

Our proposal, with respect to the work by Serrano et al. [2015], is to switch to a more generic target platform, which will provide us an easy access to powerful kernel instrumentation, as described in the previous chapter, present in most general-purpose Linux-based distributions.

To be able to conduct fine-grained energy debugging of the network stack, we must somehow isolate the network activity from the consumption of the rest of the system. To this aim, the next section is devoted to dissect and discuss the components of a laptop computer in order to understand their implications in the global power consumption, and how to minimise their impact. Then, subsequent sections explore the roots of the cross-factor.

4.1 Anatomy of a Laptop Computer

A laptop computer is a complex and power-hungry piece of hardware. It comprises a number of components, both hardware (battery, screen, hard disk drive, fan, wireless card, RAM memory, CPU) and software (services, kernel, drivers), that require a thorough discussion. Some of them are not present in other devices (e.g., the fan), and some others are essentially different in terms of performance and power requirements.

Battery The battery is a serious obstacle for energy measurements. Although using it as power source is actually possible, it is totally impractical because it prevents long term experiments, and the constant need for recharging is a waste of time. Then, the use of an external power source is highly advisable, but in this case the battery must be removed to avoid noise coming from battery charging and discharging.

Nevertheless, usually supplying DC power through the power jack socket is not enough. Most manufacturers are interested in forcing the user to acquire and use original parts. As a consequence, most laptops are capable of detecting the AC adapter and take unexpected[1] decisions. This is generally done through a third connection in the power jack. For example, in the case of Dell computers, this third wire goes to a transistor-shaped component placed in the AC transformer. Actually, this component is a small memory that can be read using a 1-wire protocol. It stores a serial number that identifies the AC adapter.

[1] That is, unexpected... for the user.

Summing up, if the laptop does not detect this memory, the BIOS can do improper[2] things. For instance, we detected that Dell computers' BIOS do not allow the OS to control CPU frequency scaling. Fortunately, it is very straightforward to *borrow* such a component from an official AC adapter and attach it permanently to the third connection of the power jack socket. In this way, any power source looks as an original part to the BIOS, and the OS kernel can freely manage all the system's capabilities.

[2] Again, improper... from the user's standpoint.

Screen Same as for smartphones[3], the screen is the most energy-hungry component in a laptop computer. It typically accounts for more than a half of the total energy consumed when the computer is just powered on and idling. Thus, the screen constitutes a very high and variable (as it depends on the GPU activity) baseline consumption that must be avoided in wireless experiments. In Linux, this can be done by finding the backlight device entry in the `/sys/class` subsystem and simply resetting it, so that the screen is permanently powered off.

[3] A. Carroll and G. Heiser. An Analysis of Power Consumption in a Smartphone. In *USENIX annual technical conference*, volume 14, pages 21–21. Boston, MA, 2010

Hard Disk Drive Regarding the system's non-volatile memory, we cannot get rid of it because it is needed for the OS storage. Commonly, laptops are equipped with hard disk drives (HDD), which are mechanical devices powered with voltages ranging from 5 to 12 V. HDDs are proven to be energy-hungry devices[4] with a consumption variability in ascending order of tens of Watts. As a consequence, every read/write during an experiment generates an intractable noise.

On the other hand, all the devices studied in Serrano et al. [2015]

[4] A. Hylick, R. Sohan, A. Rice, and B. Jones. An Analysis of Hard Drive Energy Consumption. In *2008 IEEE International Symposium on Modeling, Analysis and Simulation of Computers and Telecommunication Systems*, pages 1–10. IEEE, Sept. 2008. ISBN 978-1-4244-2817-5. DOI: 10.1109/MASCOT.2008.4770567

use flash memories. This kind of non-volatile memory is the best option, because its consumption variability is three orders of magnitude below HDD's[5]. In our experiments, we replaced the original HDD by a solid-state disk (SSD). Given that an SSD is composed of NAND flash units, its consumption is much smaller and far more stable.

[5] L. Grupp, A. Caulfield, J. Coburn, S. Swanson, E. Yaakobi, P. Siegel, and J. Wolf. Characterizing flash memory: Anomalies, observations, and applications, 2009. ISSN 1072-4451

Fan The thermal characteristics of a laptop computer require a cooling subsystem: heat sinks (CPU and GPU), air ducts and typically one fan (at least). The fan is regulated dynamically with a pulse-width modulation (PWM) technique. This component becomes an unpredictable source of electrical noise, as its operation point depends on the computer's thermal state.

Suppressing the fan is not an option because, at some point, the CPU will heat and the computer will turn off. Our solution was to set it at fixed medium speed with the help of the `i8k` kernel module and the `i8kfan` user-space application.

Wireless Card This is the last part of the wireless transmission chain and the first of the reception one. In principle, the energy models in the literature assure us that a linear behaviour is expected, independently of the manufacturer or model. However, there are a couple of factors that may lead us to select a particular card.

Supported capabilities Nowadays, it is very difficult to perform interference-free wireless experiments over ISM bands without an anechoic chamber[6]. The 2.4 GHz band is typically overcrowded, while in the 5 GHz band we have better chances to find a clear channel. Thus, an 802.11a-capable card is advisable.

[6] Especially when all your fellows work in similar research topics.

Manufacturer Distinct manufacturers (and models) have better or worse driver support in the Linux kernel. For instance, Intel PRO/Wireless cards are known for requiring a binary firmware to operate. On the other hand, Atheros released some source from their binary HAL to help the open-source community add support for their chips. As a result, there are completely free and open-source *FullMAC*[7] drivers available for all Atheros chipsets.

[7] See `https://wireless.wiki.kernel.org/en/developers/documentation/glossary` for the definition of *FullMAC* and *SoftMAC* drivers.

All our experiments were conducted using the Atheros AR9280-based 802.11a/b/g/n Half Mini-PCI Express card that was characterised in Section 3.4.1.

RAM Memory The random-access memory is a fundamental peripheral device in a computer system: it holds the instructions of the running programs —the OS kernel included— and the data associated. Therefore, our first guess was that the RAM memory could play

a meaningful role in the energy consumption of a wireless communication.

CPU The CPU is another power-hungry component. For many years, the first CPUs were like bulbs: they were consuming the same power whether they were doing something useful or not. In fact, they executed *junk code* (i.e., a loop of NOPs) during idle time. Later, CPU architects realised that more intelligent things can be done in such periods of time (for instance, enabling some kind of energy-saving mechanism).

Nowadays, CPUs are becoming more and more complex. Without seeking to be exhaustive, a modern CPU has arithmetic control units (ALUs), pipelines, a control unit, registers, several levels of cache, some clocks, etcetera. But more interestingly, modern CPUs implement several power management mechanisms (see Figure 4.1), which are covered by the Advanced Configuration and Power Interface (ACPI).

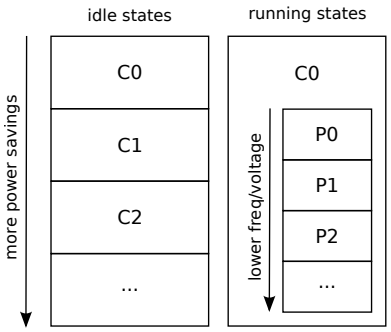


Figure 4.1: CPU P- and C-states.

P-states Also known as *frequency scaling*. When the CPU is running (i.e., executing instructions), one of these states apply. P0 means maximum power and frequency. As Px increases, the voltage is lower and, thus, the frequency of the clock, and thus the energy consumed, is scaled down.

C-states When the CPU is idle, it enters a Cx state. C0 means maximum power, because junk code is being executed. This can be a little bit confusing because, as Figure 4.1 shows, the CPU is also in C0 when it is running. In general, C0 means *the CPU is busy doing something*, whether executing actual programs (running) or something not useful (idle). In C1, the CPU is halted, and can return to an executing state almost instantaneously. In C2, the main clock is stopped and, as Cx increases, more and more functional units are shut off. As a consequence, returning from a deep C-state is very expensive in terms of latency.

For example, Intel Haswell processors support up to eight C-states: C1, C1E, C3, C6, C7s, C8, C9, C10. However, the BIOS just reports two C-states to the ACPI driver, which are called C1 and C2. We have verified, by comparing idle consumptions for each C-state, that the correspondence is as follows:

- ACPI C1 corresponds to Intel C1: the CPU is halted and stops executing instructions when it enters into idle mode.
- ACPI C2 corresponds to Intel C6: this is a new sleep state introduced in the Haswell architecture.

Finally, multicore systems introduce additional complexity, because the OS decides how many cores become active at any time, and each core has its own power management subsystem (i.e., P- and C-states). Therefore, our experiments are always carried out in single-core mode to simplify the analysis.

Services There may be a lot of active user-space services (also called *daemons*) in a Linux system by default. They can add noise to our measurements in two ways: by consuming CPU time and writing logs to disk. Hence, identifying and disabling not essential services is desirable.

Kernel There exist two power management subsystems for each CPU in the Linux kernel: `cpufreq`[8] controls P-states and `cpuidle`[9] controls C-states. Both subsystems have a similar architecture, separating mechanism (driver[10]) from policy (governor[11]).

The `cpufreq` governor has several policies that focus on certain P-states or frequencies to the detriment of others, e.g., `performance` (high frequencies) or `powersave` (low frequencies). It is also possible to manually fix certain frequency or range of frequencies.

The `cpuidle` governor takes in the next timer event as main input. Each C-state has a certain energy cost and exit latency. Thus, intuitively there are two decision factors that the governor must consider: the energy break-even point and the performance impact. The next timer event is a good predictor in many cases, but not perfect since there are other sources of wake-ups (e.g., interrupts). Therefore, it computes a correction factor using an array of 12 independent factors through a running average. Moreover, it is possible to manually disable the C-states (excepting C0).

[8] `https://www.kernel.org/doc/Documentation/cpu-freq`

[9] `https://www.kernel.org/doc/Documentation/cpuidle`

[10] It provides the platform-dependent state detection capability and the mechanisms to support entry/exit into/from different states. By default, there exists an ACPI driver that implements standard APIs. Usually, CPU-specific drivers are capable of detecting more states than ACPI-compliant ones.

[11] It is an algorithm that takes in several system parameters as input and decides the state to activate.

Drivers All Linux drivers are compiled as separate modules. In particular, wireless drivers[12], along with the entire 802.11 subsystem, can be compiled out-of-tree within the *backports* project[13]. This is very useful in order to use the latest drivers on older kernels.

The wireless driver module interacts directly with the network interface controller (NIC). In our case, the selected card uses the `ath9k` driver. The function `ath9k_tx()` is the entry point for the transmission path. The driver fills the transmission descriptors, copies the buffer into the NIC memory and sets up several registers that trigger the transmission.

[12] `http://wireless.kernel.org`

[13] `https://backports.wiki.kernel.org`

In order to conduct energy breakdowns like the one depicted in Figure 3.5, Serrano et al. [2015] claim that their methodology discards

packets *right after the driver*. This statement becomes uncertain when a closer look at any wireless driver is taken. According to our experiments, discarding a frame *before* the buffer is copied into the NIC implies that *only* half of the driver is actually taken into account for the cross-factor value. On the other hand, if we try to discard it in the very last instruction of the driver (i.e., avoiding setting the register that triggers the hardware transmission), then the module crashes if the NIC's memory is not cleaned up, a non-trivial task that anyway would consume more energy. Due to this, and combined to the fact that drivers differ greatly from one to another, we do not include the driver into the definition of cross-factor, given the difficulty of isolating driver and NIC consumptions.

As with the variety of drivers mentioned above, a similar argument can be wield against the user-space consumption. Therefore and from here on, we define *kernel cross-factor* as the energy consumed from the system call that delivers the message until the driver is reached. *Cross-factor*, as is, maintains the original definition given by Serrano et al. [2015] to avoid confusion.

We would also like to highlight that our way of interrupting the transmission path by discarding a frame in the driver is a bit different from Serrano et al. [2015]. They conducted this breakdown by commenting a driver function. This method implies the need for recompiling the driver, which is time-consuming and not very portable[14].

For our part, we generate packets with a short string at the beginning of packet payloads. The presence of this *magic string* triggers the packet drop. Our method, despite introducing a very small overhead, is agile and portable (for instance, it can be implemented on the fly using SystemTap).

[14] Think, for instance, about a similar task for a function contained in the kernel core.

THE LAPTOP COMPUTER selected for our experiments is a Dell Latitude E5540 with Intel Core i5-4300U CPU at 1.9 GHz and 8 GB of SODIMM DDR3 RAM at 1.6 GHz, equipped with an Atheros AR9280-based 802.11a/b/g/n Half Mini-PCI Express card. In our measurements, we use Fedora-default pre-compiled kernels. We arranged two separate partitions: one with Fedora 12 and kernel version 2.6.32 and the other with Fedora 20 and kernel 3.14. Only the latter supports Intel-specific drivers, thus we use ACPI drivers only in order to operate under similar conditions.

4.2 *Cross-Factor: Separating the Wheat from the Chaff*

We have identified the two main components suspected of being responsible of the cross-factor: CPU and RAM memory. Our first

priority is to isolate and quantify the impact of the RAM. It is not possible to regulate its activity, but this can be done for the CPU. For this purpose, the CPU was fixed at Po-Co states, i.e., always running at maximum frequency, maximum energy consumption. We performed an energy breakdown using this configuration in kernel 3.14 and the results are shown in Figure 4.2.

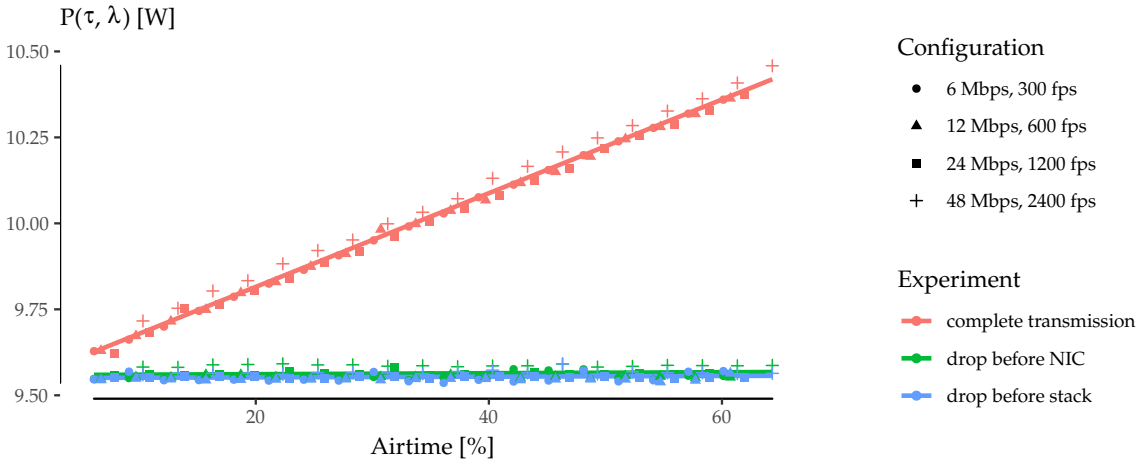


Figure 4.2: Energy breakdown with the CPU fixed at Po-Co states.

The lines appear superimposed: the laptop is consuming the same power among different parameters, different packet rates. Hence, one important conclusion to be drawn is that the RAM memory has no significant impact in the overall energy consumption of wireless transmissions. The noise can be ascribed to the fact that not all the instructions consume exactly the same energy[15]. Other possible sources of noise are cache and pipeline flushes.

With this simple experiment, we have demonstrated that the CPU is the leading cause of cross-factor in laptops, and it is clear that the `cpuidle` subsystem has a central role, because a CPU spends most of the time in idle mode[16]. From now on, and in order to take a deeper look at C-states, we remove a variable by keeping the P-state fixed at Po (maximum frequency).

4.3 *Power Consumption in Unattended Idle Mode*

The Soekris net4826-48 used in Section 3.3.1 is equipped with an AMD Geode SC1100 CPU that supports ACPI C1, C2 and C3 states[17]. Unfortunately, it seems that Linux distributions for embed devices, such as Voyage Linux, disable `cpuidle` in their kernels, which means that the OS has no control over the idle mode. In such conditions, we know now that the CPU cannot be in Co all the time, because the device does not consume the same power with different parameters.

[15] V. Tiwari, S. Malik, A. Wolfe, and M. Tien-Chien Lee. Instruction level power analysis and optimization of software. *Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology*, 13(2-3):223–238, Aug. 1996. ISSN 0922-5773. DOI: 10.1007/BF01130407

[16] L. A. Barroso and U. Hölzle. The Case for Energy-Proportional Computing. *Computer*, 40(12):33–37, Dec. 2007. ISSN 0018-9162. DOI: 10.1109/MC.2007.443

[17] `http://datasheets.chipdb.org/upload/National/SC1100.pdf`

What is happening then?

BACK TO OUR LAPTOP, it is possible to disable `cpuidle` through the kernel command-line. The idle power consumption in this situation, which we call *unattended idle mode*, reveals that the laptop is entering C1. This fact can be extrapolated to the Soekris case, which makes sense, since there is no governor to resolve which C-state is the more suitable at any given time. Thus, the processor simply halts when there is no work to do.

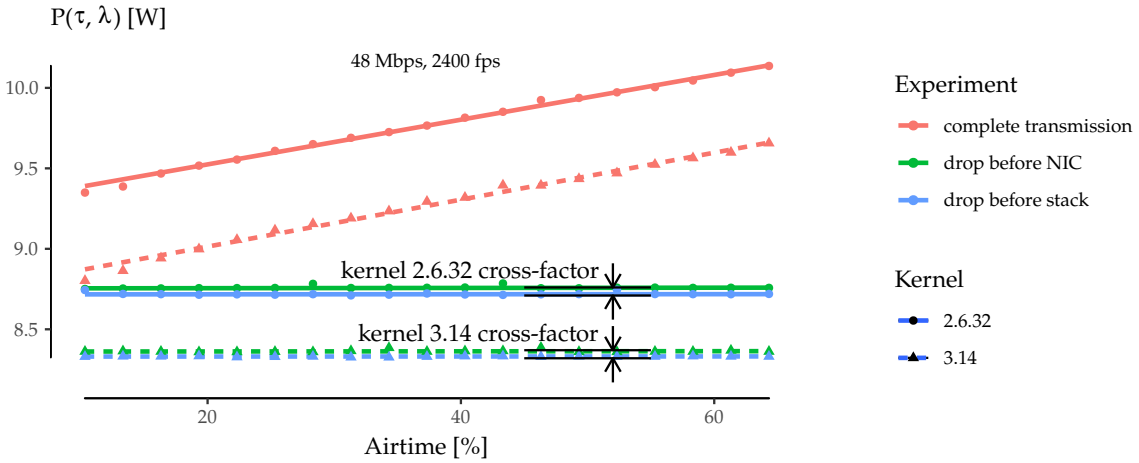


Figure 4.3: Power consumption breakdown vs. airtime with fixed C1 state for kernels 2.6.32 and 3.14.

Figure 4.3 shows the energy breakdown for both kernels, 2.6.32 and 3.14, when only the C1 state is enabled. As can be seen, the obtained kernel cross-factor is almost negligible, which suggests that Intel Haswell's C1 state saves a very small amount of power, unlike the Soekris' C1 state as shown in Figure 3.5.

There is also a baseline power difference between kernels. This offset can be ascribed to several factors. For instance, a lot of code has changed —and probably improved— between those kernel versions. In particular, the scheduler and the `cpuidle` algorithms have evolved. Moreover, the compiler used has changed also.

At this respect, we can calculate the complete cross-factor (including the user-space, as done in Section 3.3.1) by extracting the slopes of the regressions of Figure 4.4. These values are comparable to the Linksys case reported by Serrano et al. [2015]: 0.51(2) mJ (kernel 2.6.32) and 0.38(2) mJ (kernel 3.14).

It is also important to note that, unlike the results from Figure 3.5, there is absolutely no dependence on the frame size in this case. Our guess is that RAM memory consumption would be proportional to the frame size and may have a small but still perceptible impact in low-power devices, but it is negligible compared to the consumption

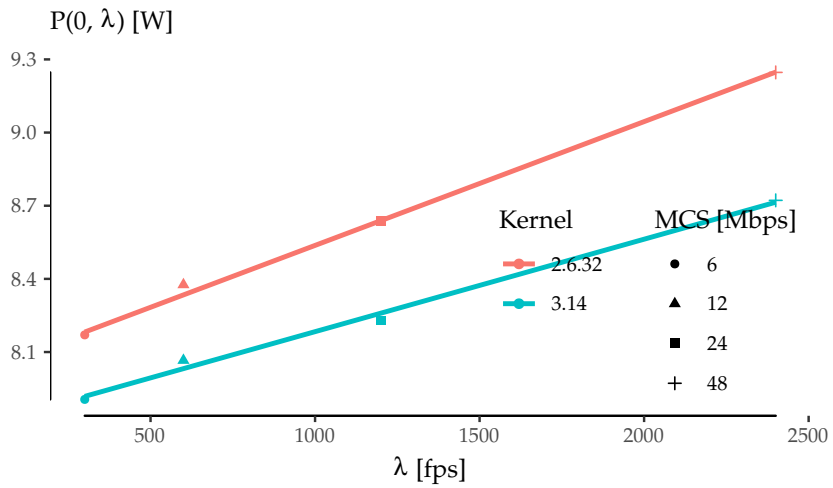


Figure 4.4: Power consumption offset ($\tau = 0$) vs. framerate with fixed C1 state for kernels 2.6.32 and 3.14.

of a laptop's CPU. As a consequence, the frame size can be removed as a parameter from the cross-factor analysis in laptops.

4.4 *Power Consumption with Full `cpuidle` Subsystem*

With the knowledge acquired so far, we can move onto a more realistic scenario by enabling the whole `cpuidle` subsystem, i.e., keeping both ACPI C-states enabled and letting the governor decide.

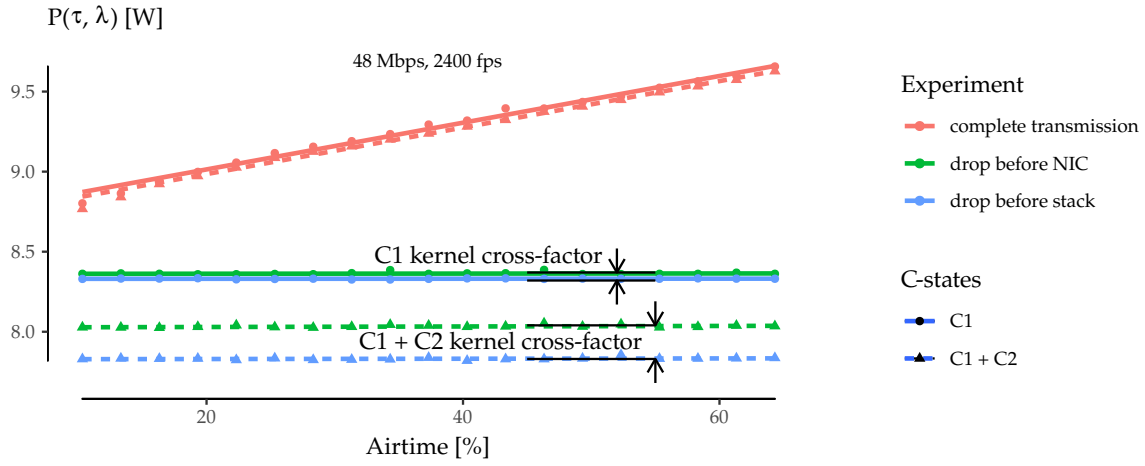


Figure 4.5: Power consumption breakdown vs. airtime with two `cpuidle` configurations for kernel 3.14.

Figure 4.5 depicts the energy breakdown for kernel 3.14 with full `cpuidle` subsystem (C1+C2 enabled) and compares it to the previous case (C1 only). By enabling C2, the consumption appears to be always lower up to driver level (blue and green lines). Nevertheless, the consumption of complete transmissions (red lines) is lower in the 300 fps case (not shown here), but it is the same in the 2400 fps case.

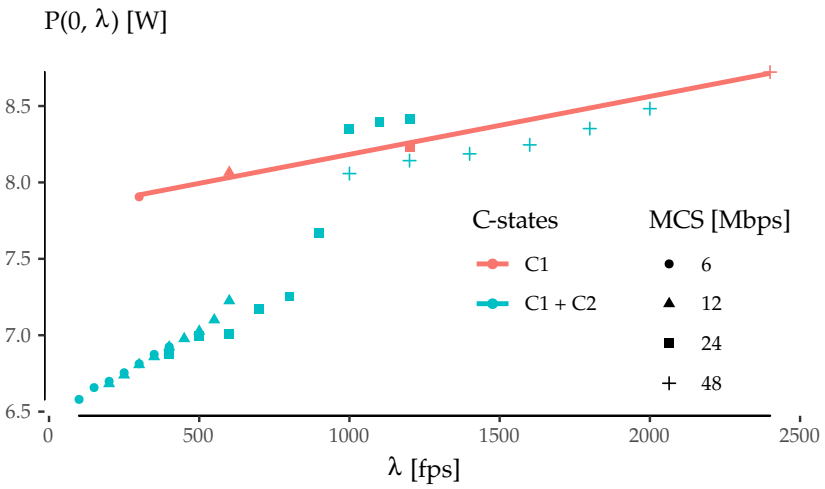


Figure 4.6: Power consumption offset ($\tau = 0$) vs. framerate with two `cpuidle` configurations for kernel 3.14.

Figure 4.6 compares the offsets of complete transmissions in Figure 4.5 for both cases: C1+C2 and C1 only. The red line corresponds to C1: as expected, its behaviour is linear as seen in Figure 4.4. On the other hand, the C1+C2 case (blue points) is not linear globally. It comprises three clearly distinct parts: when the framerate is low, there is an approximately linear behaviour because the CPU only uses C2; when the framerate is high, C2 is no longer used, and the slope matches the red line; between them, the behaviour becomes unpredictable because of the mix of C1 and C2. Therefore, the cross-factor as defined by Serrano et al. [2015] makes no sense anymore. When all the C-states are active, there is no linear behaviour anymore: we cannot talk neither about a slope nor a fixed energy toll per frame.

Furthermore, we had assumed, as Serrano et al. [2015], that we can simply drop the packets at certain points, measure the mean power up to those points and represent all this as an energy breakdown. But obviously this is not true either. For instance, Figure 4.5 shows that the CPU is not entering C2 when complete transmissions are performed, and the consumption is the same as in the C1-only case. On the other hand, the CPU is clearly spending some time in C2 when the frames are dropped earlier. Even it seems that the network stack is consuming more power because the energy gap (between green and blue lines) is larger. Evidently, it should be the opposite: the stack would be consuming less power as soon as it enters a lower C-state.

4.5 *Exploring the `cpuidle` Subsystem*

As stated in previous sections, the `cpuidle` subsystem is a very complex component. Kernel timer events are the main input for the governor algorithm as they often indicate the next wake-up of the CPU, but the running average used to scale the latter makes it unpredictable, since it depends on the recent state of the whole machine. The purpose of this section is to shed some light on the linkage between the residence time of C-states, the number of wake-ups per second, the CPU load and the transmission of wireless frames.

We implemented a very simple application[18] with two modes of operation: it is capable of setting a kernel timer at a given constant rate and, when this timer is triggered, it (i) does nothing or (ii) sends a UDP packet. At the same time, it calculates the mean residence time of each C-state over the whole execution.

[18] Available at `https://github.com/Enchufa2/udperf`

Figures 4.7 have been compiled using this tool. The additional CPU load was added on top of the latter using a modified version of `lookbusy`[19]. Figure 4.8 compares the two previous figures in terms of power consumption.

[19] `http://www.devin.com/lookbusy`

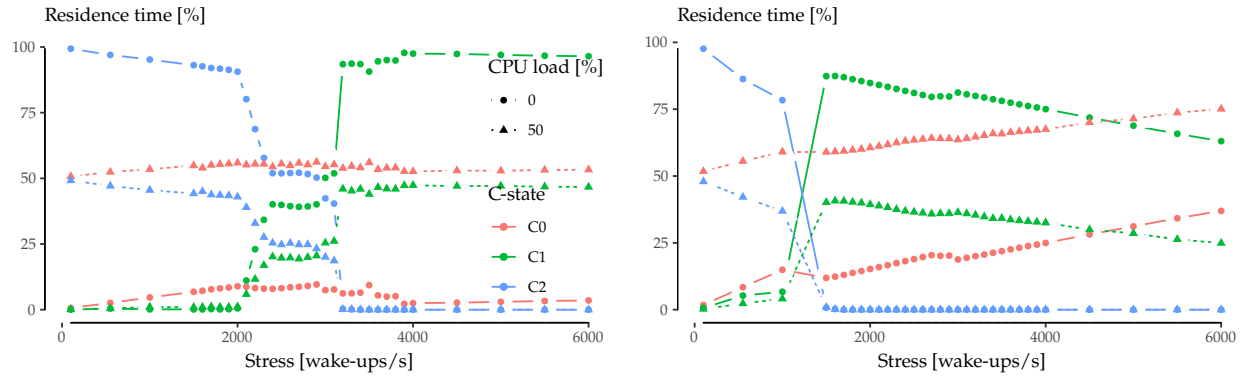


Figure 4.7: Residence time of each C-state vs. wake-ups/s for kernel 3.14. Each wake-up does nothing (left) or performs a UDP transmission (right).

In Figure 4.7 (left), the only source of wake-ups is the kernel timer that our tool sets. Each C-state is represented by a different colour, and shapes and line types distinguish between CPU loads. The first observation is that the addition of a substantial source of CPU load has no impact on the distribution of residence times. Another important observation is that, up to 2000 and from 3500 wake-ups/s onwards, there is only one active idle state (C2 or C1 respectively), and the behaviour is linear. This fact can be verified by checking the power consumption (Figure 4.8, red lines). From 2000 to 3500 wake-ups/s, the transition between C-states occurs in a non-linear way.

In Figure 4.7 (right), on other hand, there is another source of wake-ups: hardware interrupts caused by the wireless card each time a packet is sent. The transition between states occurs earlier because

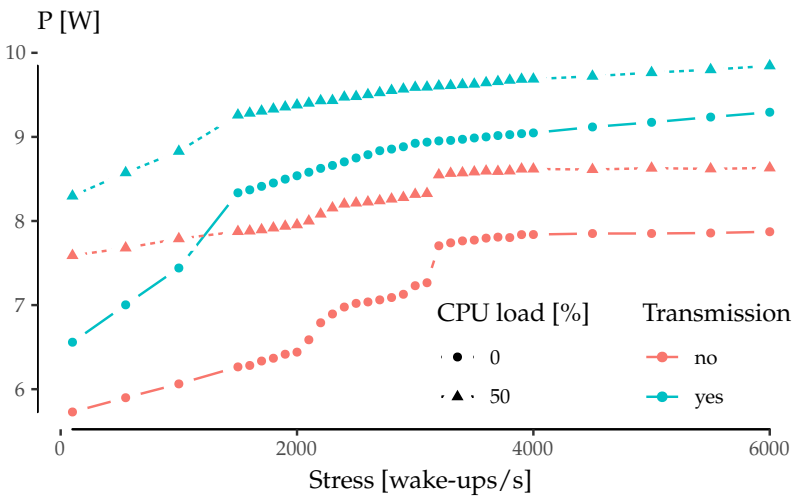


Figure 4.8: Power consumption offset vs. wake-ups/s for kernel 3.14.

there is actually twice the number of wake-ups. And, again, the CPU load shows no impact on the distribution of residence times.

These are partial results and are limited to constant rate wake-ups, but these findings are in line with the non-linearities previously discovered in the cross-factor and they confirms the enormous complexity we face.

4.6 Summary

This chapter follows the path set out by Serrano et al. [2015] with the discovery of the cross-factor, an energy toll not accounted by classical energy models and associated to the very fact that frames are processed along the network stack. We have introduced the laptop as a more suitable device to perform whole-device energy measurements in order to deseed the root causes of the cross-factor by taking advantage of the wide range of debugging tools that such platform enables.

Our results[20], albeit preliminary, provide several fundamental insights on this matter:

- We have identified the CPU as the leading cause of the cross-factor in laptops. Thus, the cross-factor shows absolutely no dependence on the frame size, because the RAM memory has no significant impact in the overall energy consumption of wireless transmissions. On the other hand, low-powered devices, like the Soekris, show a very small but perceptible dependency that can be ascribed to the RAM memory.
- The CPU's C-state management plays a central role in the en-

[20] **I. Ucar**, A. Azcorra, and A. Banchs. Deseeding Energy Consumption of Network Stacks. In *6th Annual International IMDEA Networks Workshop*, June 2014. (invited poster); and **I. Ucar** and A. Azcorra. Deseeding energy consumption of network stacks. In *IEEE 1st International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI)*, pages 7–16, Sept. 2015. ISBN 978-1-4673-8167-3. DOI: 10.1109/RTSI.2015.7325085

ergy consumption, because a CPU spends most of the time in idle mode.

- When the C-state management subsystem is not present in the OS, the device enters C1 in idle mode (halted) and cannot benefit from lower idle states.
- In contrast to low-powered devices, the C1 state of a laptop's CPU saves a very small amount of power.
- With a fully functional C-state management subsystem, the linear behaviour disappears. In consequence, we cannot talk about cross-factor as a fixed energy toll per frame.
- A non-linear behaviour implies that we cannot perform energy breakdowns by dropping packets inside the transmission chain. Therefore, new methodologies and techniques are required to enable energy debugging.
- C-state residence times depend primarily on the number of wake-ups per second produced by software and hardware interrupts. However, they show no dependence on the CPU load.

Further research is needed in order to fully understand the key role of the C-state subsystem in the energy consumption of wireless communications, as well as to investigate other processor capabilities not accounted for in this work, such as P-states and multicore support.

5 Leveraging Micro-Sleep Opportunities in 802.11

In this chapter, we revisit the 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. Then, we analyse 802.11 to identify potential micro-sleep opportunities, taking into account practical CSMA-related issues (e.g., capture effect, hidden nodes) not considered in prior work.

Building on this knowledge, we design μNap, a local standard-compliant energy-saving mechanism for 802.11 WLANs. With μNap, a station 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.

Finally, the performance of the algorithm is evaluated based on our measurements and real wireless traces. In a brief discussion of the impact and applicability of our mechanism, we draw attention to the need for standardising hardware capabilities in terms of energy in 802.11.

5.1 State Transition Times in 802.11 Cards

From the hardware point of view, the standard Power Save (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[1]. In fact, other types of devices, such as microcontroller-based devices or modern general-purpose CPUs, implement sleep states in the same way.

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

[1] X. Zhang and K. G. Shin. E-MiLi: Energy-Minimizing Idle Listening in Wireless Networks. *IEEE Transactions on Mobile Computing*, 11(9):1441–1454, Sept. 2012. ISSN 1536-1233. DOI: 10.1109/TMC.2012.112

$\Delta t_{\text{sleep}} = t_{\text{on}} - t_{\text{off}}$, 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

$$E_{\text{save}} = (P_{\text{idle}} - P_{\text{sleep}}) \cdot \Delta t_{\text{sleep}} \tag{5.1}$$

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, Figure 5.1 presents a conceptual breakdown of a generic micro-sleep.

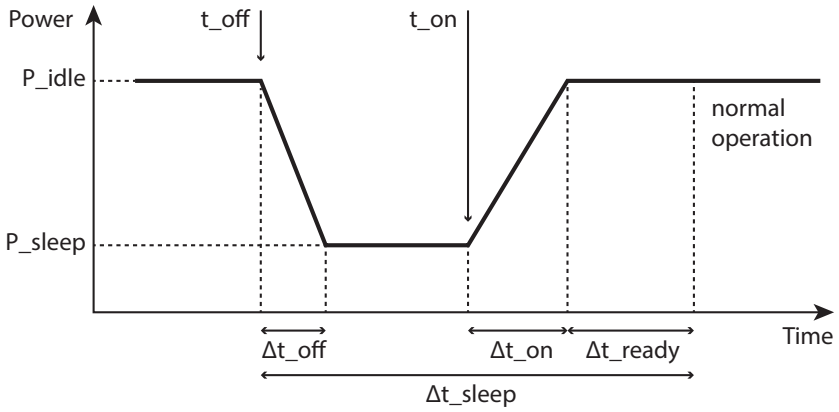


Figure 5.1: Generic sleep breakdown.

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:

$$\begin{aligned} E'_{\text{save}} &= E_{\text{save}} - E_{\text{waste}} \\ &= (P_{\text{idle}} - P_{\text{sleep}}) \cdot (\Delta t_{\text{sleep}} - \Delta t_{\text{ready}}) \\ &\quad - \int_{\Delta t_{\text{off}} \cup \Delta t_{\text{on}}} (P - P_{\text{sleep}}) \cdot dt \end{aligned} \tag{5.2}$$

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:

$$\Delta t_{\text{sleep,min}} = \Delta t_{\text{off}} + \Delta t_{\text{on}} + \Delta t_{\text{ready}} \tag{5.3}$$

Performing this experimental characterisation requires the ability to timely trigger the sleep mode on demand. As stated in Section 4.1, most COTS cards are not suitable for this task, because they implement all the low-level operations in an internal proprietary binary firmware. However, those cards based on the open-source driver `ath9k`, like the one presented in previous chapters[2] are well suited for our needs. The driver has access to very low level functionality (e.g., supporting triggering the sleep mode by just writing into a register).

[2] Atheros AR9280.

This characterisation follows the setup depicted in Section 3.4, Figure 3.7. The card under test is associated to an access point (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 will be able to measure the limiting parameters $\Delta t_{\text{off}}, \Delta t_{\text{on}}, \Delta t_{\text{ready}}$.

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. A simple hack in the `ath9k` module[3] allows us to perform the following experiment:

[3] Available at `https://github.com/Enchufa2/crap/tree/master/ath9k/downup`.

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. 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 Figure 5.2 (left).

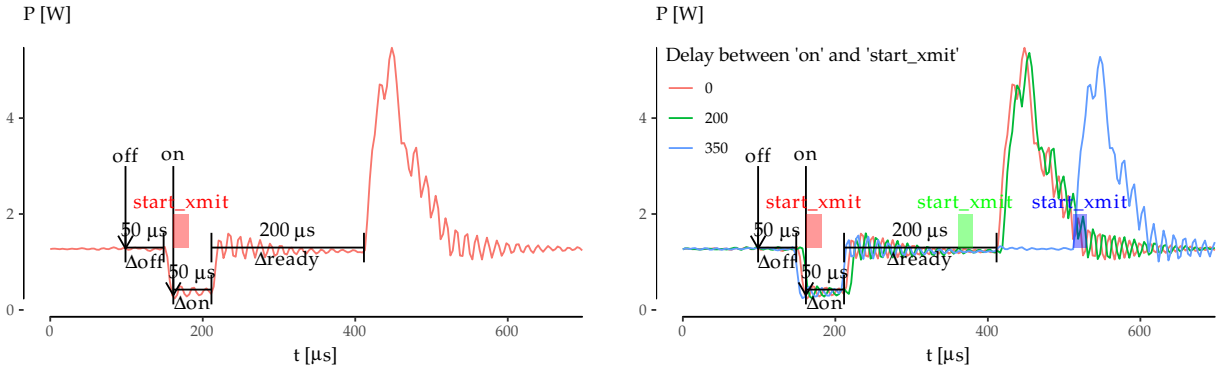


Figure 5.2: Atheros AR9280 timing characterisation.

As we can see, the card spends $\Delta t_{off} = 50$ 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 $\Delta t_{on} = 50$ 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. Figure 5.2 (right) 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 $\Delta t_{ready} = 200$ μs to get ready to transmit/receive after returning to idle.

Summing 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 $\Delta t_{sleep,min} = 300$ 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 $\Delta t_{\text{on}} + \Delta t_{\text{ready}} = 250\ \mu$s before the end of the sleep. Note also that the card wastes a fixed time Δt_{waste} consuming P_{idle}:

$$\Delta t_{\text{waste}} = \Delta t_{\text{off}} + \Delta t_{\text{ready}} \tag{5.4}$$

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 Equation (5.2) can be simplified as follows:

$$E_{\text{waste}} \approx (P_{\text{idle}} - P_{\text{sleep}}) \cdot \Delta t_{\text{waste}} \tag{5.5}$$

5.2 *Protocol Analysis and Practical Issues*

The key idea of this chapter 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, such a mechanism should be local in order to be incrementally deployable, standard-compliant, and should take into account real-world practical issues. For this purpose, we first identify potential micro-sleep opportunities in 802.11, and explore well-known practical issues of WLAN networks that had not been addressed by previous energy-saving schemes.

5.2.1 *Identifying Potential Micro-Sleep Opportunities*

Due to the CSMA mechanism, an 802.11 station (STA) receives every single frame from its Service Set Identifier (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). 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. 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)[4].

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[5]. 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[6]. A non-zero

[4] For instance, this could be the ACK following a data frame or the CTS + data + ACK following an RTS.

[5] Which is 32 768; see IEEE [2012a, Table 8-3] for further details about the duration/ID field encoding

[6] See IEEE [2012a, Table 8-105].

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.

5.2.2 *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 Lee et al. [2007][7], 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, Lee et al. [2007] suggests that a high-power transmission could blind a low-power one *at any time*, even when the actual data

[7] J. Lee, W. Kim, S.-J. Lee, D. Jo, J. Ryu, T. Kwon, and Y. Choi. An experimental study on the capture effect in 802.11a networks. In *Proceedings of the the second ACM international workshop on Wireless network testbeds, experimental evaluation and characterization - WinTECH '07*, page 19, New York, New York, USA, Sept. 2007. ACM Press. ISBN 9781595937384. DOI: 10.1145/1287767.1287772

transmission has begun. This is called *Message in Message* (MIM) in the literature[8], 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.

[8] N. Santhapuri, R. R. Choudhury, J. Manweiler, S. Nelakuduti, S. Sen, and K. Munagala. Message in message mim: A case for reordering transmissions in wireless networks. In *In HotNets*, 2008; and W. Wang, W. K. Leong, and 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*, WiNTECH '14, pages 41–48, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-3072-5. DOI: 10.1145/2643230.2643231

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[9]. 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.

[9] F. Gringoli and L. Nava. OpenFWWF - Open FirmWare for WiFi networks, 2015

Figure 5.3 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.

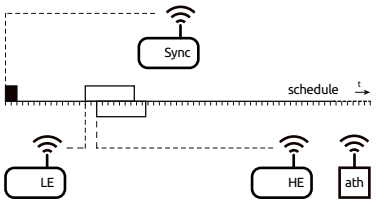


Figure 5.3: Measurement setup for the MIM effect.

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 transmission 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 5.1. When the energy gap is small (≤ 5 dB), the MIM effect never enters into play as we can

Table 5.1: Message-in-message effect.

		LE frames		HE frames	
ΔP [dB]	Δt [μs]	% 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
	≥ 48	99.10	99.75	0.34	0.00
	≥ 144	98.94	0.00	97.32	0.00
≥ 35	0	0.18	0.00	99.37	0.00
	16	0.37	1.11	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
	≥ 144	95.33	0.00	99.64	0.00

see from the first part of Table 5.1. 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 (i.e., more than 35 dB), then the behaviour of the QCA card changes radically as we can see from the second part of Table 5.1: 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.

5.2.3 *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. [2010][10] and Prasad et al. [2014][11] 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.

[10] B. Balaji, B. R. Tamma, and B. S. Manoj. A Novel Power Saving Strategy for Greening IEEE 802.11 Based Wireless Networks. In *2010 IEEE Global Telecommunications Conference GLOBECOM 2010*, pages 1–5. IEEE, Dec. 2010. ISBN 978-1-4244-5636-9. DOI: 10.1109/GLOCOM.2010.5684071

[11] R. Prasad, A. Kumar, R. Bhatia, and 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)*, pages 1–6. IEEE, Jan. 2014. ISBN 978-1-4799-2317-5. DOI: 10.1109/CONECCT.2014.6740349

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 inter-

preted 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[12].

Henceforth, we draw the following conclusions:

- 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.

[12] Note that this frame might be received because of the MIM effect explained previously.

- 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:

$$p_{e,b} \approx 1 - (1 - p_b)^{15} \tag{5.6}$$

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:

$$p_{eg,b}(i) = p_{e,b} \cdot \frac{15 - \mathrm{Hw}(i)}{15} \tag{5.7}$$

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 in real networks, we have analysed the SIGCOMM'08 data set[13] and gathered which duration values are the most common. In the light of the results depicted in Table 5.2, 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

[13] A. Schulman, D. Levin, and N. Spring. CRAWDAD data set umd/sigcomm2008 (v. 2009-03-02), Mar. 2009. URL http://crawdad.org/umd/sigcomm2008/

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

Table 5.2: Most frequent duration values.

to an excessive sleep interval using a single-bit error model is the following:

$$p_{\text{loss}} = p_{eg,b} \approx p_{e,b} \approx 1 - (1 - \text{BER})^{15} \tag{5.8}$$

This analysis assumes independent errors. However, it is well known that 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[14], which has been successfully applied in telecommunications to describe burst error distributions[15]. 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[16], which is our case.

[14] J. Neyman. On a new class of 'contagious' distributions, applicable in entomology and bacteriology. *The Annals of Mathematical Statistics*, 10(1): 35–57, 1939

[15] E.g., by ITU-R [2005], Becam et al. [1985] and Irvin [1991].

[16] B. Cornaglia and M. Spini. Letter: New statistical model for burst error distribution. *European transactions on telecommunications*, 7(3):267–272, 1996

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

$$p_N(k) = \frac{\lambda_b^k}{k!} e^{-\lambda_B} \sum_{i=0}^{\infty} \frac{i^k}{i!} \lambda_B^i e^{-i\lambda_b} \tag{5.9}$$

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 can be transformed into a recursive formula with finite sums:

$$\begin{aligned} p_N(k) &= \frac{\lambda_B \lambda_b e^{-\lambda_b}}{k} \sum_{j=0}^{k-1} \frac{\lambda_b^j}{j!} p_N(k-1-j) \\ p_N(0) &= e^{-\lambda_B\left(1-e^{-\lambda_b}\right)} \end{aligned} \tag{5.10}$$

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:

$$p_{\text{loss}} = \sum_{k=1}^{15} p_{15}(k) \tag{5.11}$$

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

Figure 5.4 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 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.

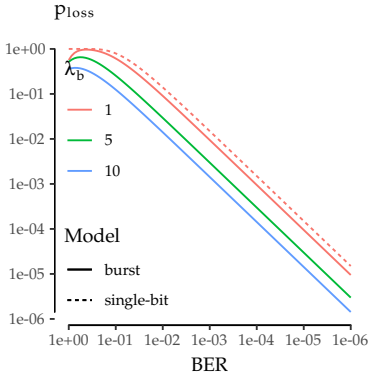


Figure 5.4: Frame loss probability given a BER level.

5.3 *µNap Algorithm*

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 5.1, 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 Figure 5.5, 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

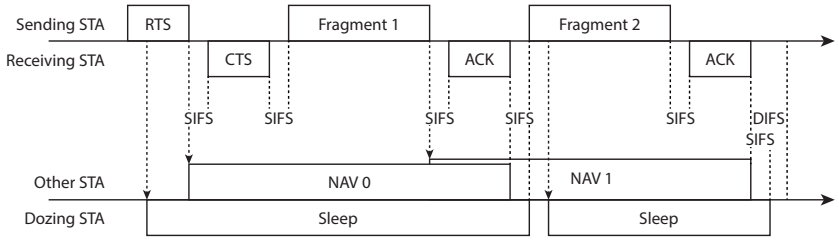


Figure 5.5: RTS/CTS-based fragmented transmission example and μNap's behaviour.

trigger another sleep. This implies that the STA can doze for an additional SIFS, as Figure 5.5 shows, and wait in idle state until a DIFS is completed before trying to contend again.

BASED ON THE ABOVE, Algorithm 5.6 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[17]. 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 (`Set_Sleep` in Algorithm 5.6). 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.

[17] D. S. Berger, F. Gringoli, N. Facchi, I. Martinovic, and 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*, WiSec '14, pages 105–116, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2972-9. DOI: 10.1145/2627393.2627403

```
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
```

Figure 5.6: μNap implementation. Main loop modification to leverage micro-sleeps.

5.4 Performance Evaluation

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

5.4.1 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 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 SIGCOMM'08 data set[18]. 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.

[18] A. Schulman, D. Levin, and N. Spring. CRAWDAD data set umd/sigcomm2008 (v. 2009-03-02), Mar. 2009. URL http://crawdad.org/umd/sigcomm2008/

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 take into account the 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 our mechanism). 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.

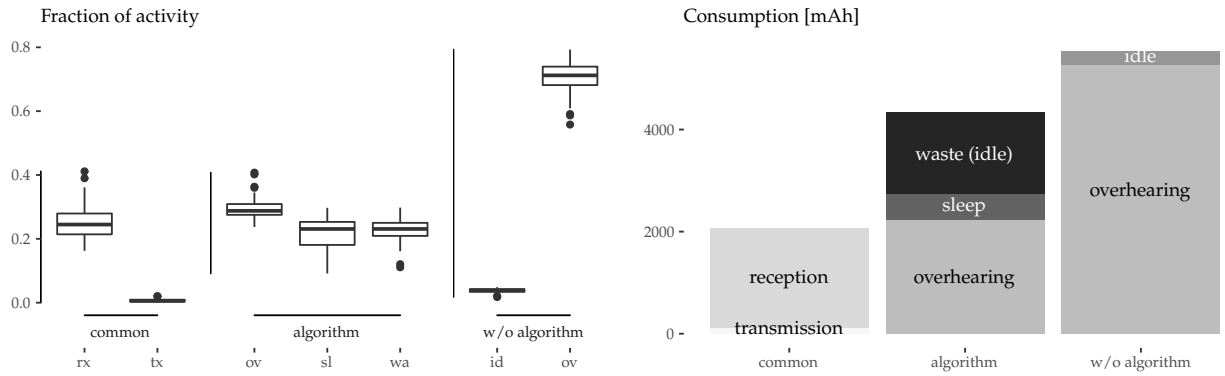


Figure 5.7: Normalised activity aggregation (left) and energy consumption aggregation (right) of all STAs.

The activity aggregation of all STAs is normalised and represented in Figure 5.7 (left). 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 Section 3.4.1. Figure 5.7 (right) 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 (i) prevents the card from going to sleep when the overhearing time is not sufficiently long, and (ii) 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.

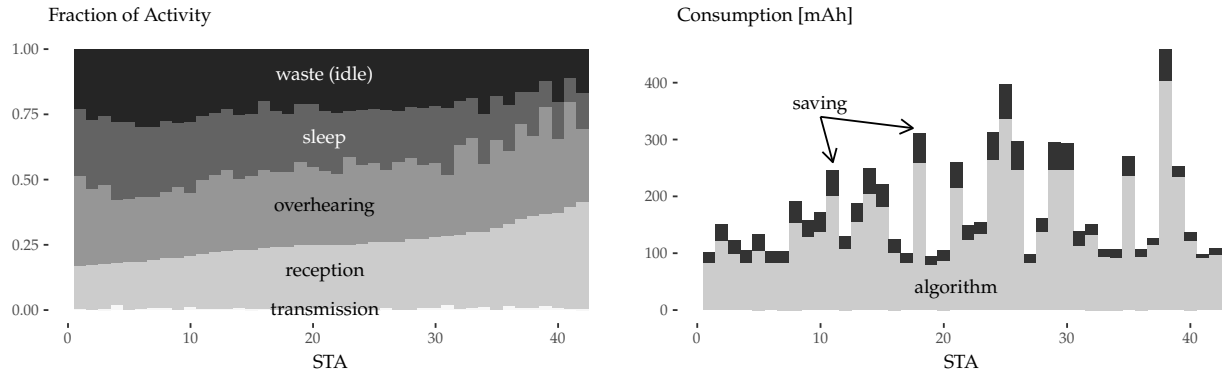


Figure 5.8: Normalised activity (left) and energy consumption (right) per STA.

Figure 5.8 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.

5.4.2 *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

$$\frac{E'_{\text{save}}}{E_{\text{save}}} = \frac{E_{\text{save}} - E_{\text{waste}}}{E_{\text{save}}} \approx 1 - \frac{\Delta t_{\text{waste}}}{\Delta t_{\text{sleep}}} \tag{5.12}$$

which results from the combination of Equations (5.1), (5.2) and (5.5).

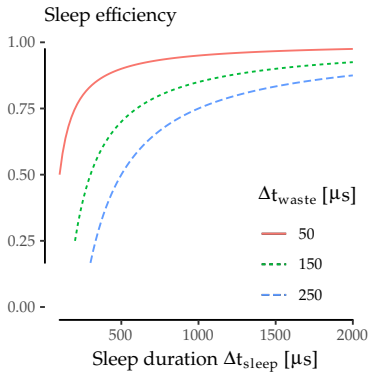


Figure 5.9: Sleep efficiency $E'_{\text{save}}/E_{\text{save}}$ as Δt_{waste} decreases.

Figure 5.9 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.

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,

$$\Delta t_{\text{sleep,min}} \leq \Delta t_{\text{DATA}} + \Delta t_{\text{SIFS}} + \Delta t_{\text{ACK}} + \Delta t_{\text{SIFS}} \tag{5.13}$$

and expanding the right side of the inequality,

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

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}, Figure 5.10 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.

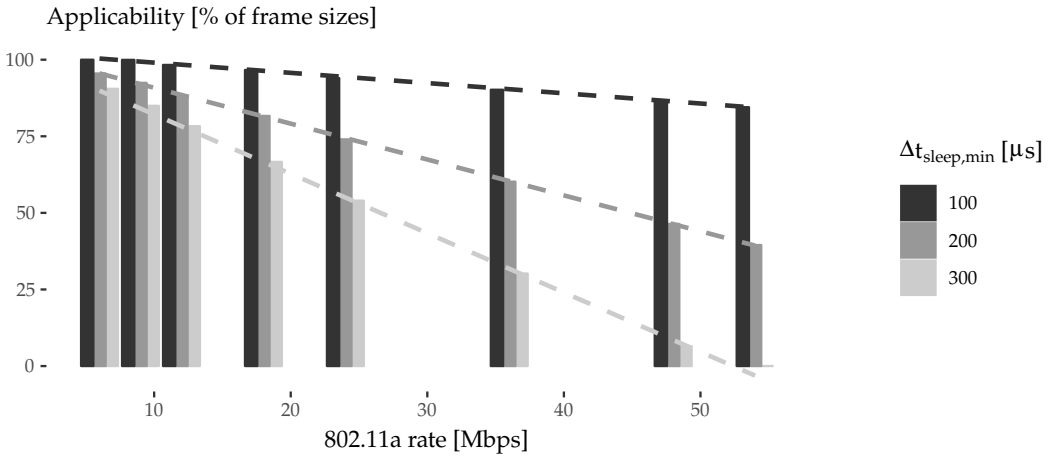


Figure 5.10: Algorithm applicability for common transmissions ($\leq$ 1500 bytes + ACK) in 802.11a DCF mode.

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 saving/restoring the state). Secondly, Figure 5.2 (left) indicates that a transmission is far more aggressive, in terms of a sudden power rise, than a return from sleep. From this standpoint, $\Delta t_{\text{ready}} = 200\ \mu$s 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[19] of the PHYs, which are in the range of tens of μs in the worst case[20].

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 on 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.

[19] E.g., $\Delta t_{\text{w}_{\text{phy}}}$ would be equivalent to our $\Delta t_{\text{on}} + \Delta t_{\text{ready}}$.

[20] See IEEE [2012b, Table 78-4]

5.5 *Summary*

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[21] 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.

[21] A. Azcorra, **I. Ucar**, A. Banchs, F. Gringoli, and P. Serrano. μNap: Practical micro-sleeps for 802.11 WLANs. *Computer Communications*, 110:175–186, Sept. 2017. ISSN 0140-3664. DOI: 10.1016/j.comcom.2017.06.008; and A. Azcorra, **I. Ucar**, A. Banchs, F. Gringoli, and P. Serrano. Energy-saving method based on micro-shutdowns for a wireless device in a telecommunications network. WO/2018/015601, Jan. 2018. URL http://www.google.com/patents/WO2018015601

Part II Mathematical Modelling

6 Rate Adaptation and Power Control in 802.11

Rate adaptation is a fundamental technique in 802.11 networks by which wireless transmistters adapt their transmission data rate to the prevailing channel conditions. As we have learnt from previous chapters, the energy consumption of wireless interfaces depends on the chosen modulation and coding scheme, which defines the data rate. On the other hand, *transmission power control*, which has gained increase attention due to network densification, tries to minimise the amount of radiated power (and, indirectly, the energy consumption) given a set of radio conditions.

These two competing techniques are typically assessed in terms of throughput achieved, while it is assumed that optimality in terms of this performance figure implies optimality in terms of energy efficiency. This chapter tackles the latter question from a formal standpoint. For this purpose, and we first present a joint goodput (i.e., the throughtput delivered on top of 802.11) and energy consumption model. Building on this model, we numerically study the relationship between energy consumption and throughput performance in 802.11.

6.1 Joint Goodput-Energy Model

With the aim of isolating the variables of interest, namely, modulation and coding scheme (MCS) and transmission power (TXP), in the following we develop a joint goodput-energy model for a single 802.11 spatial stream in the absence of interfering traffic. This model will let us delve into the fundamental relationship between goodput and energy consumption optimality without confounding effects such as collisions or MIMO.

Beyond this primary intent, it is worth noting that these assumptions conform with real-world scenarios in the scope of recent trends in the IEEE 802.11 standard development (e.g., amendments 11ac and 11ad), where device-to-device communications (mainly through beamforming and MU-MIMO) are of paramount importance.

We base our study on the work by Qiao et al. [2002][1], which develops a robust goodput model that meets the established requirements. This model analyses the IEEE 802.11a Distributed Coordination Function (DCF) over the assumption of additive white Gaussian noise (AWGN) channel without interfering traffic.

[1] D. Qiao, S. Choi, and K. Shin. Goodput analysis and link adaptation for IEEE 802.11a wireless LANs. *Mobile Computing, IEEE Transactions on*, 1(4): 278–292, Oct 2002. ISSN 1536-1233. DOI: 10.1109/TMC.2002.1175541

LET US BRIEFLY INTRODUCE the reader to the main concepts, essential to our analysis, of the goodput model by Qiao et al. [2002]. Given a packet of length l ready to be sent, a frame retry limit $n_{\max}$ and a set of channel conditions $\hat{s} = \{s_1, \ldots, s_{n_{\max}}\}$ and modulations $\hat{m} = \{m_1, \ldots, m_{n_{\max}}\}$ used during the potential transmission attempts, the *expected effective goodput* $\mathcal{G}$ is modelled as the ratio between the expected delivered data payload and the expected transmission time as follows:

$$\mathcal{G}(l, \hat{s}, \hat{m}) = \frac{\mathbb{E}\left[\text{data}\right]}{\mathbb{E}\left[\mathcal{D}_{\text{data}}\right]} = \frac{\Pr[\text{succ} \mid l, \hat{s}, \hat{m}] \cdot l}{\mathbb{E}\left[\mathcal{D}_{\text{data}}\right]} \tag{6.1}$$

where $\Pr[\text{succ} \mid l, \hat{s}, \hat{m}]$ is the probability of successful transmission conditioned to $l, \hat{s}, \hat{m}$, given in Qiao et al. [2002, Equation (5)]. This model is valid as long as the coherence time is equal or greater than a single retry, i.e., the channel condition s_i is constant.

The expected transmission time is defined as follows:

$$\begin{aligned}\mathbb{E}\left[\mathcal{D}_{\text{data}}\right] = {} & (1 - \Pr[\text{succ} \mid l, \hat{s}, \hat{m}]) \cdot \mathcal{D}_{\text{fail}|l,\hat{s},\hat{m}} \\ & + \Pr[\text{succ} \mid l, \hat{s}, \hat{m}] \cdot \mathcal{D}_{\text{succ}|l,\hat{s},\hat{m}}\end{aligned} \tag{6.2}$$

where

$$\begin{aligned}\mathcal{D}_{\text{succ}|l,\hat{s},\hat{m}} = \sum_{n=1}^{n_{\max}} \Pr[n \text{ succ} \mid l, \hat{s}, \hat{m}] \cdot \Bigg\{ & \overline{T}_{\text{bkoff}}(1) \\ & + \sum_{i=2}^{n_{\max}} \left[\overline{T}_{\text{bkoff}}(i) + T_{\text{data}}(l, m_i) + \overline{\mathcal{D}}_{\text{wait}}(i)\right] \\ & + T_{\text{data}}(l, m_1) + T_{\text{SIFS}} + T_{\text{ACK}}(m'_n) + T_{\text{DIFS}} \Bigg\}\end{aligned} \tag{6.3}$$

is the average duration of a successful transmission and

$$\mathcal{D}_{\text{fail}|l,\hat{s},\hat{m}} = \sum_{i=1}^{n_{\max}} \left[\overline{T}_{\text{bkoff}}(i) + T_{\text{data}}(l, m_i) + \overline{\mathcal{D}}_{\text{wait}}(i+1)\right] \tag{6.4}$$

is the average time wasted during the $n_{\max}$ attempts when the transmission fails.

$\Pr[n \text{ succ} \mid l, \hat{s}, \hat{m}]$ is the probability of successful transmission at the n-th attempt conditioned to $l, \hat{s}, \hat{m}$, and $\overline{\mathcal{D}}_{\text{wait}}(i)$ is the average waiting time before the i-th attempt. Their expressions are given in Qiao et al. [2002, Equations (7)–(8)]. The transmission time (T_{data}), ACK time (T_{ACK}) and average backoff time ($\overline{T}_{\text{bkoff}}$) are given in Qiao et al. [2002, Equations (1)–(3)]. Finally, T_{SIFS} and T_{DIFS} are 802.11a parameters, and they can be found also in Qiao et al. [2002, Table 2].

THE SELECTED ENERGY MODEL is again the work by Serrano et al. [2015][2], which stands as the most accurate energy model for 802.11 devices published so far. While classical models focused on the wireless interface solely, this one demonstrates empirically that the energy consumed by the device itself cannot be neglected as a device-dependent constant contribution. Conversely, devices incur an energy cost derived from the frame processing, which may impact the relationship that we want to evaluate in this paper.

[2] P. Serrano, A. Garcia-Saavedra, G. Bianchi, A. Banchs, and A. Azcorra. Per-Frame Energy Consumption in 802.11 Devices and Its Implication on Modeling and Design. *IEEE/ACM Transactions on Networking*, 23(4):1243–1256, Aug 2015. ISSN 1063-6692. DOI: 10.1109/TNET.2014.2322262

PUTTING TOGETHER BOTH MODELS, we are now in a position to build a joint goodput-energy model for 802.11a DCF. Let us consider the average durations (6.3) and (6.4). Based on their expressions, we multiply the idle time ($\overline{\mathcal{D}}_{\text{wait}}$, $\overline{T}_{\text{bkoff}}$, T_{SIFS}, T_{DIFS}) by ρ_{id}, the transmission time (T_{data}) by ρ_{tx}, and the reception time (T_{ACK}) by ρ_{rx}. The resulting expressions are the average energy consumed in a successful transmission $\mathcal{E}_{\text{succ}|l,\hat{s},\hat{m}}$ and the average energy wasted when a transmission fails $\mathcal{E}_{\text{fail}|l,\hat{s},\hat{m}}$:

$$\begin{aligned}\mathcal{E}_{\text{succ}|l,\hat{s},\hat{m}} = \sum_{n=1}^{n_{\max}} \Pr[n \text{ succ} \mid l,\hat{s},\hat{m}] \cdot \Big\{ & \rho_{\text{id}} \overline{T}_{\text{bkoff}}(1) \\ & + \sum_{i=2}^{n_{\max}} \left[\rho_{\text{id}} \overline{T}_{\text{bkoff}}(i) + \rho_{\text{tx}} T_{\text{data}}(l, m_i) + \rho_{\text{id}} \overline{\mathcal{D}}_{\text{wait}}(i)\right] \\ & + \rho_{\text{tx}} T_{\text{data}}(l, m_1) + \rho_{\text{id}} T_{\text{SIFS}} + \rho_{\text{rx}} T_{\text{ACK}}(m'_n) + \rho_{\text{id}} T_{\text{DIFS}} \Big\}\end{aligned} \tag{6.5}$$

$$\mathcal{E}_{\text{fail}|l,\hat{s},\hat{m}} = \sum_{i=1}^{n_{\max}} \left[\rho_{\text{id}} \overline{T}_{\text{bkoff}}(i) + \rho_{\text{tx}} T_{\text{data}}(l, m_i) + \rho_{\text{id}} \overline{\mathcal{D}}_{\text{wait}}(i+1)\right] \tag{6.6}$$

Then, by analogy with (6.2), the *expected energy consumed per frame transmitted*, $\mathbb{E}[\mathcal{E}_{\text{data}}]$, can be written as follows:

$$\begin{aligned}\mathbb{E}[\mathcal{E}_{\text{data}}] = \gamma_{\text{xg}} + (1 - \Pr[\text{succ} \mid l,\hat{s},\hat{m}]) \cdot \mathcal{E}_{\text{fail}|l,\hat{s},\hat{m}} \\ + \Pr[\text{succ} \mid l,\hat{s},\hat{m}] \cdot \mathcal{E}_{\text{succ}|l,\hat{s},\hat{m}}\end{aligned} \tag{6.7}$$

It is noteworthy that the receiving cross-factor does not appear in this expression because ACKs (acknowledgements) are processed in the network card exclusively, and thus its processing toll is negligible.

Finally, we define the *expected effective energy efficiency* μ as the ratio between the expected delivered data payload and the expected energy consumed per frame, which can be expressed in *bits per Joule* (bpJ):

$$\mu(l,\hat{s},\hat{m}) = \frac{\mathbb{E}[\text{data}]}{\mathbb{E}[\mathcal{E}_{\text{data}}]} \tag{6.8}$$

6.2 *Numerical Results*

Building on the joint model presented in the previous section, here we explore the relationship between optimal goodput and energy efficiency in 802.11a. More specifically, our objective is to understand the behaviour of the energy efficiency of a single spatial stream as the MCS and TXP change following our model to meet the optimal goodput.

6.2.1 *Optimal Goodput*

We note that the main goal of RA, generally, is to maximise the effective goodput that a station can achieve by varying the parameters of the interface. In terms of the model discussed in the previous section, a rate adaptation algorithm would aspire to fit the following curve:

$$\max \mathcal{G}(l, \hat{s}, \hat{m}) \tag{6.9}$$

We provide the numerical results for this goodput maximisation problem in Figure 6.1, which are in good agreement with those obtained in Qiao et al. [2002]. For the sake of simplicity but without loss of generality we fix $l = 1500$ octets and $n_{\max} = 7$ retries, and assume that the channel conditions and the transmission strategy are constant across retries ($\hat{s} = \{s_1, \ldots, s_1\}$ and $\hat{m} = \{m_1, \ldots, m_1\}$).

Figure 6.1 illustrates which mode (see Table 6.1) is optimal in terms of goodput, given an SNR level. We next address the question of whether this optimisation is aligned with energy efficiency maximisation.

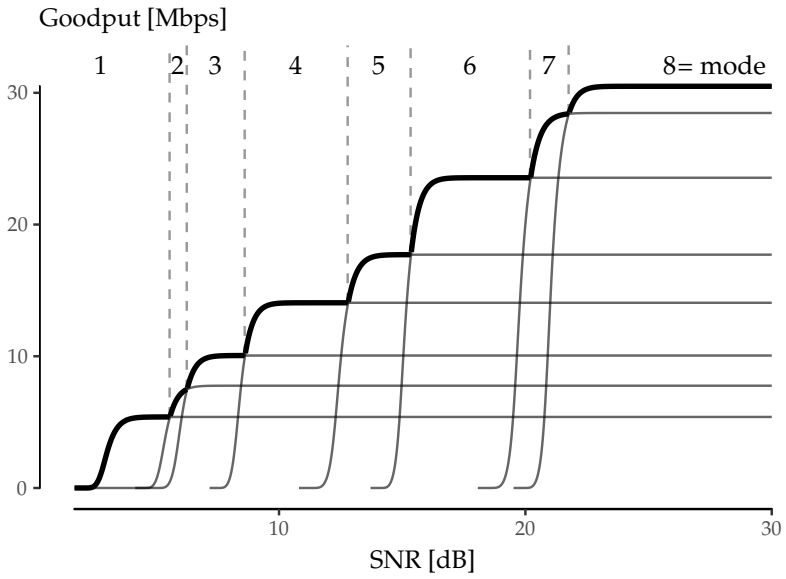


Figure 6.1: Optimal goodput (bold envelope) as a function of SNR.

Mode Index	MCS [Mbps]
1	6
2	9
3	12
4	18
5	24
6	36
7	48
8	54

Table 6.1: IEEE 802.11a PHY.

6.2.2 *Extension of the Energy Parametrisation*

The next step is to delve into the energy consumption of wireless devices. Serrano et al. [2015] provides real measurements for five devices: three AP-like platforms (Linksys WRT54G, Raspberry Pi and Soekris net4826-48) and two hand-held devices (HTC Legend and Samsung Galaxy Note 10.1). Two of the four parameters needed are constant (ρ_{id}, γ_{xg}), and the other two (ρ_{tx}, ρ_{rx}) depend on the MCS and the TXP used. However, the characterisation done by Serrano et al. [2015] is performed for a subset of the MCS and TXP available, so we next detail how we extend the model to account for a larger set of operation parameters.

A detailed analysis of the numerical figures presented in Serrano et al. [2015] suggests that ρ_{rx} depends linearly on the MCS, and that ρ_{tx} depends linearly on the MCS and the TXP (in mW). Based on these observations, we define the following linear models:

$$\begin{aligned} \rho_{tx} &= \alpha_0 + \alpha_1 \cdot \text{MCS} + \alpha_2 \cdot \text{TXP} \\ \rho_{rx} &= \beta_0 + \beta_1 \cdot \text{MCS} \end{aligned} \tag{6.10}$$

The models are fed with the data reported in Serrano et al. [2015], and the resulting fitting is illustrated in Figure 6.2, while Table 6.2 collects the model estimates for each device (with errors between parentheses), as well as the adjusted r-squared. Since these linear models show a very good fit, they support the generation of synthetic data for the different MCS and TXP required.

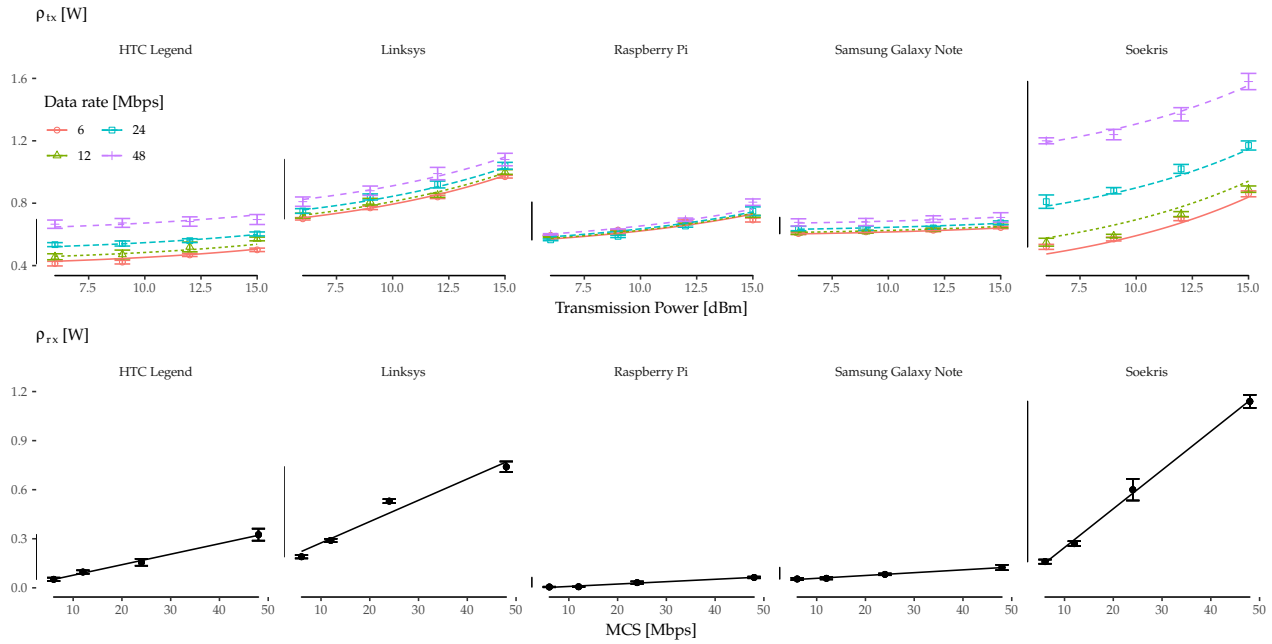


Figure 6.2: Linear regressions. ρ_{tx} fit as a function of MCS and TXP (top) and ρ_{rx} fit as a function of MCS (bottom).

	ρ_{tx} model estimates				ρ_{rx} model estimates		
Device	α_0 [W]	α_1 [W/Mbps]	α_2 [W/mW]	adj. r^2	β_0 [W]	β_1 [W/Mbps]	adj. r^2
HTC Legend	0.35(1)	0.0052(3)	0.021(3)	0.966	0.013(3)	0.0064(1)	0.995
Linksys	0.54(1)	0.0028(2)	0.075(3)	0.982	0.14(2)	0.0130(7)	0.957
Raspberry Pi	0.48(2)	0.0008(4)	0.044(5)	0.866	-0.006(1)	0.00146(5)	0.982
Samsung Galaxy Note	0.572(4)	0.00166(8)	0.0105(9)	0.975	0.041(1)	0.00173(4)	0.993
Soekris	0.17(3)	0.0170(6)	0.101(7)	0.986	0.010(8)	0.0237(3)	0.998

Table 6.2: Linear regressions.

6.2.3 *Energy Consumption*

To compute the energy consumption using the above parametrisation, first we have to define the assumptions for the considered scenario. We assume for simplicity a device-to-device communication, with fixed and reciprocal channel conditions during a sufficient period of time (i.e., low or no mobility). As we have discussed before, our primary goal is to isolate MCS and TXP as variables of interest, but we must not forget that these are also reasonable assumptions in scenarios targeted by recent 802.11 standard developments (11ac, 11ad).

For instance, given channel state information from a receiver, the transmitter may decide to increase the TXP in order to increase the receiver's SNR (and thus the expected goodput), or to decrease it if the channel quality is high enough. Although the actual relationship between TXP and SNR depends on the specific channel model (e.g., distance, obstacles, noise), without loss of generality, we choose a noise floor of $N = -85$ dBm in an office scenario with a link distance of $d = 18$ m in order to explore numerically the whole range of SNR while using reasonable values of TXP. The ITU model for indoor attenuation[3] gives a path loss of $L \approx 85$ dBm. Then, we can use (6.7) to obtain the expected energy consumed per frame and MCS mode, with TXP being directly related to the SNR level.

[3] ITU-R. P.1238: Propagation data and prediction methods for the planning of indoor radiocommunication systems and radio local area networks in the frequency range 300 MHz to 100 GHz. Recommendation P.1238, International Telecommunication Union, July, 2015

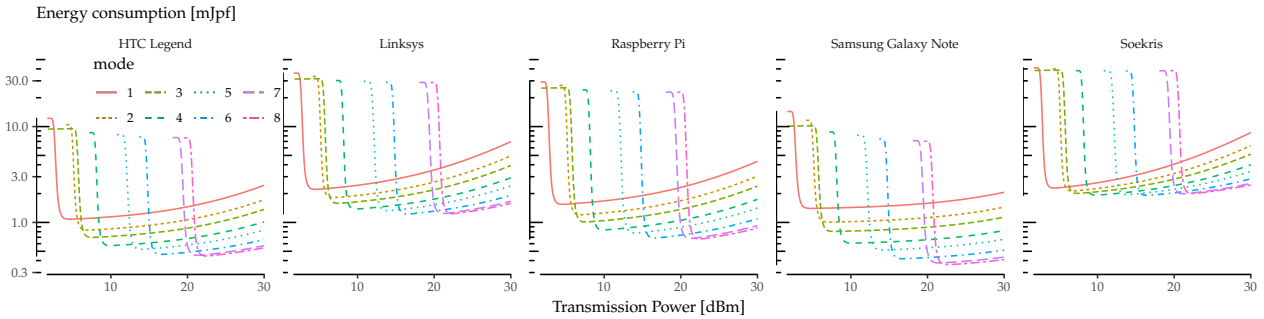


Figure 6.3: Expected energy consumption per frame in *millijoules per frame* (mJpf) under fixed channel conditions.

The results are reported in Figure 6.3. As the figure illustrates,

consumption first falls abruptly as the TXP increases for all modes, which is caused when the SNR reaches a sharp threshold level such that the number of retransmissions changes from 6 to 0 (i.e., no frame is discarded). From this threshold on, the consumption increases with TXP because, although the number of retransmissions is 0, the wireless interface consumes more power. We note that the actual value of the TXP when the consumption drops depends on the specifics of the scenario considered, but the qualitative conclusions hold for a variety of scenarios.

6.2.4 *Energy Efficiency vs. Optimal Goodput*

We can finally merge previous numerical analyses and confront energy efficiency, given by (6.8), and optimal goodput, given by (6.9), for all devices and under the aforementioned assumptions. To this aim, we plot in the same figure the energy efficiency for the configuration that maximises goodput given an SNR value vs. the obtained goodput, with the results being depicted in Figure 6.4. We next discuss the main findings from the figure.

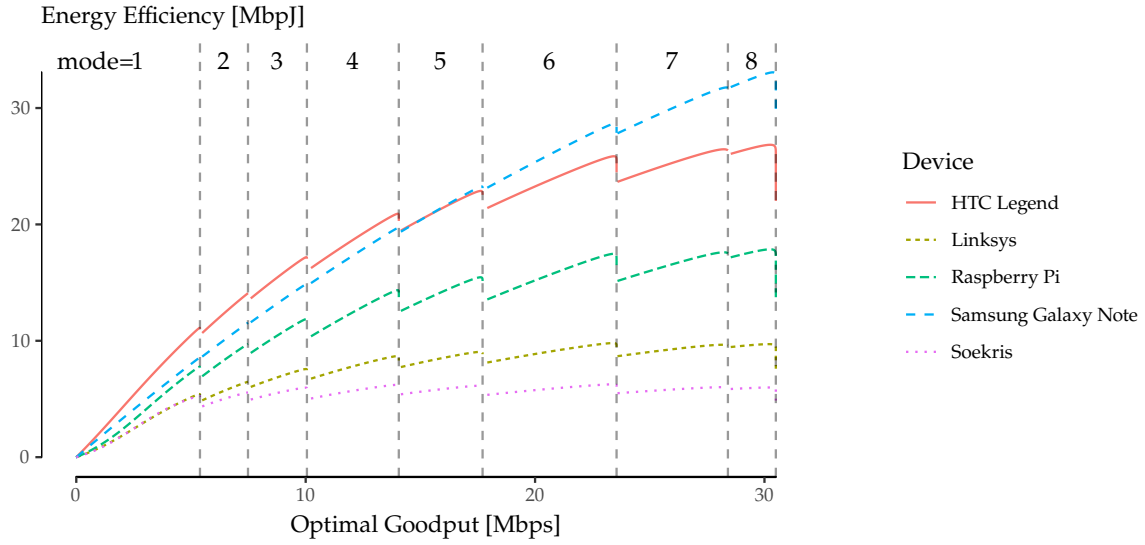


Figure 6.4: Energy efficiency vs. optimal goodput under fixed channel conditions.

First of all, we can see that the energy efficiency grows sub-linearly with the optimal goodput (the optimal goodput for each SNR value) in all cases. We may distinguish three different cases in terms of energy efficiency: high (Samsung Galaxy Note and HTC Legend), medium (Raspberry Pi) and low energy efficiency (Linksys and Soekris). Furthermore, for the case of the Soekris, we note that the "central modes" (namely, 4 and 5) are more efficient in their optimal region than the subsequent ones.

Another finding (more relevant perhaps) is that it becomes evident

that increasing the goodput does not always improve the energy efficiency: there are more or less drastic leaps, depending on the device, between mode transitions. From the transmitter point of view, in the described scenario, this can be read as follows: we may increase the TXP to increase the SNR, but if the optimal goodput entails a mode transition, the energy efficiency may be affected.

As a conclusion, we have demonstrated that optimal goodput and energy efficiency do not go hand in hand, even in a single spatial stream, in 802.11[4]. There is a trade-off in some circumstances that current rate adaptation algorithms cannot take into account, as they are oblivious to the energy consumption characteristic of the device.

[4] These findings have been further confirmed with an experimental validation, which is covered in Appendix B.

6.3 *Discussion*

6.3.1 *Sensitivity to Energy Parameter Scaling*

We next explore how the different energy parameters affect the energy efficiency vs. optimal goodput relationship. For this purpose, we selected the Raspberry Pi curve from Figure 6.4 (results are analogous with the other devices) and we scale up and down, one at a time, the four energy parameters ρ_{id}, ρ_{tx}, ρ_{rx}, and γ_{xg}. The scaling up and down is done by multiplying and dividing by 3, respectively, the numerical value of the considered parameter. One of the first results is that the impact of ρ_{rx} is negligible, a result somehow expected as the cost of receiving the ACK is practically zero. From this point on, we do not further consider this parameter.

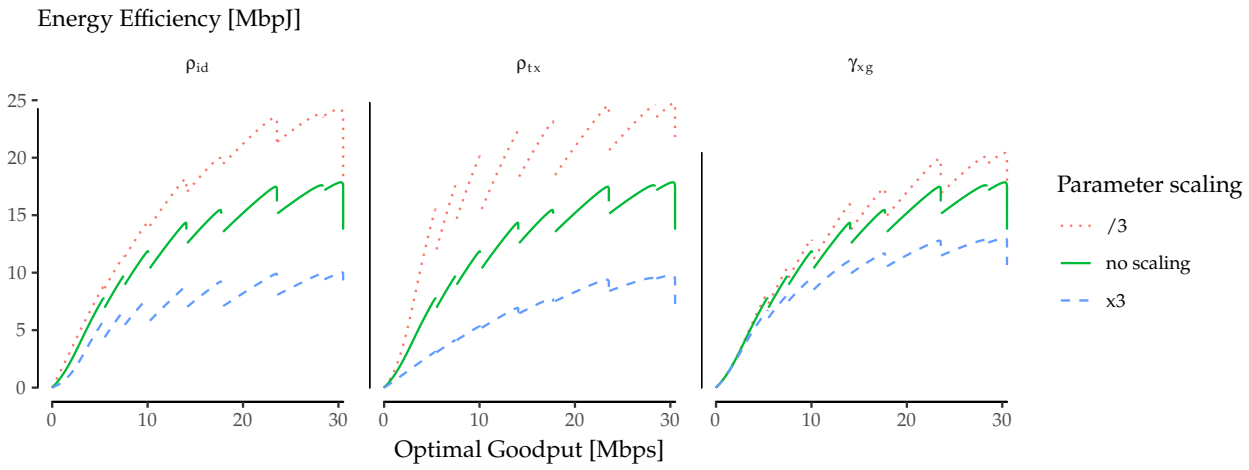


Figure 6.5: Impact of energy parameter scaling on the energy efficiency. Overall effect.

We show in Figure 6.5 the overall effect of this parameter scaling. The solid line represents the base case with no scaling (same curve as in Figure 6.4), and in dashed and dotted lines the corresponding parameter was multiplied or divided by a factor of 3, respectively.

As expected, larger parameters contribute to lower the overall energy efficiency. However, the impact on the energy efficiency drops between mode transitions is far from being obvious, as in some cases transitions are more subtle while in others they become more abrupt.

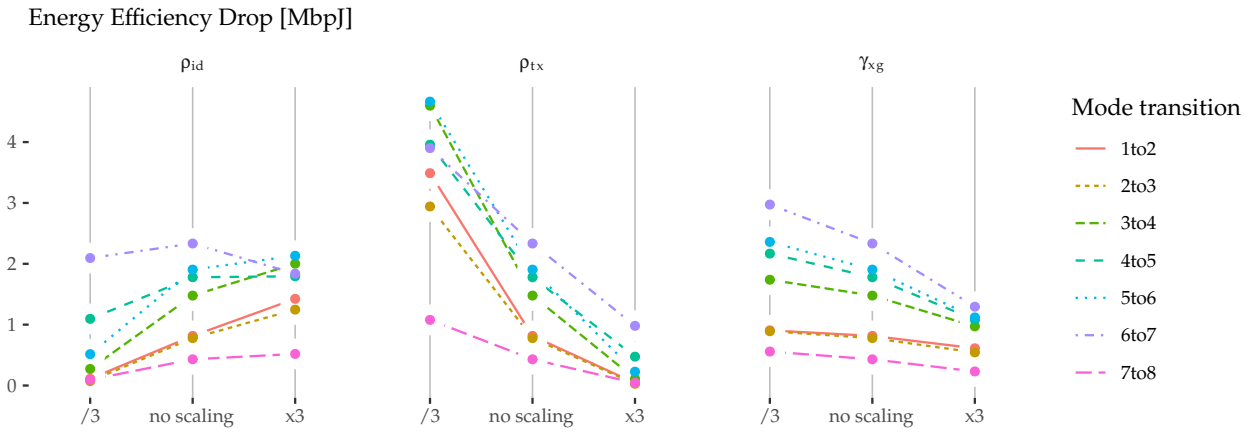


Figure 6.6: Impact of energy parameter scaling on the energy efficiency. Impact on mode transitions.

To delve into these transitions, we illustrate in Figure 6.6 the "drop" in energy efficiency when changing between modes. As it can be seen, the cross-factor γ_{xg} is the less sensitive parameter of the three, because its overall effect is limited and, more importantly, it scales all the leaps between mode transitions homogeneously. This means that a higher or lower cross-factor, which resides almost entirely in the device and not in the wireless card, does not alter the energy efficiency vs. optimal goodput relationship[5]. Thus, the cross-factor is not relevant from the RA-TPC point of view, and energy-aware RA-TPC algorithms can be implemented by leveraging energy parameters local to the wireless card.

On the other hand, ρ_{id} and ρ_{tx} have a larger overall effect, plus an inhomogeneous and, in general, opposite impact on mode transitions. While a larger ρ_{id} contributes to larger leaps, for the case of ρ_{tx}, the larger energy efficiency drops occur with smaller values of that parameter. Still, the reason behind this behaviour is the same for both cases: the wireless card spends more time in idle (and less time transmitting) when a transition to the next mode occurs, which has a higher data rate.

This effect is also evident if we compare the Samsung Galaxy Note and the HTC Legend curves in Figure 6.4. Both devices have ρ_{id} and ρ_{tx} in the same order of magnitude, but the HTC Legend has a larger ρ_{id} and a smaller ρ_{tx}. The combined outcome is a more dramatic sub-linear behaviour and an increased energy efficiency drop between mode transitions.

[5] Note that this parameter results in a constant term in Equation (6.7).

6.3.2 *Heuristics for RA-TPC Algorithms*

We have seen that the energy efficiency vs. optimal goodput relationship shows a signature "sawtooth" pattern when RA and TPC are considered for a single 802.11 spatial stream. This sawtooth shape presents a growing trend in the central part of each mode, but there are energy efficiency drops between mode transitions, which conceal a trade-off.

Parameter scaling has diverse effects on the final consumption signature, but overall, the qualitative behaviour (i.e., the shape) remains the same. The cross-factor produces an homogeneous scaling of the sawtooth. Thus, a first conclusion is that the trade-off depends on the energy parameters local to the wireless card, which means that a properly designed energy-aware RA-TPC algorithm can be device-agnostic.

Moreover, an energy-aware RA-TCP algorithm may also be card-agnostic. This is because the inefficiencies are always constrained at mode transitions, which are exactly the points at which RA-TPC algorithms take decisions. Therefore, there is no need of knowing the exact energy parametrisation, nor the instantaneous power consumption of the wireless card, in order to take energy-efficient decisions.

An RA-TPC algorithm moves along the sawtooth shapes of Figure 6.4 in two directions, namely, "up" (towards higher throughput) and "down" (towards lower throughput). In this way, an algorithm requires different policies to make a decision:

(i) In the *upwards* direction, in which mode transitions take place by increasing MCS and TXP (to achieve more goodput), a sensitive policy would be to remain in the left side of the leaps, to prevent falling into the efficiency gaps, until the link is good enough to move to a higher MCS with at least the same efficiency. An heuristic for the *upwards* policy may be the following: whenever an algorithm chooses a higher MCS, it may hold the decision for some time and, if it persists, then trigger the MCS change[6].

(ii) In the *downwards* direction, in which mode transitions take place by decreasing MCS and TXP, a sensitive policy would be to try to reach the left side of the leaps as soon as possible. However, it should be noted that this *downwards* policy is much more challenging, as it implies predicting quality drops to trigger early MCS/TXP changes.

[6] However, if this delay is too long, the algorithm will incur in inefficiencies, too.

In summary, our results suggest that one of the key points of an energy-aware RA-TPC algorithm is the management of mode transitions. A good algorithm should be *conservative* at mode transitions, in the sense that it should prefer a lower MCS and TXP until a higher MCS can be guaranteed.

6.4 Summary

We have revisited 802.11 rate adaptation and transmission power control by taking energy consumption into account. While some previous studies pointed out that MIMO rate adaptation is not energy efficient, we have demonstrated through numerical analysis that, even for single spatial streams without interfering traffic, energy consumption and throughput performance are different optimisation objectives.

Our findings[7] show that this trade-off emerges at certain "mode transitions" when maximising the goodput, suggesting that small goodput degradations may lead to energy efficiency gains. For instance, a station at the edge of a mode transition may decide to reduce the transmission power a little in order to downgrade the modulation coding scheme. Or an opportunity to achieve a better goodput by increasing the transmission power and modulation coding scheme could be delayed if the expected gain is small.

[7] **I. Ucar**, C. Donato, P. Serrano, A. Garcia-Saavedra, A. Azcorra, and A. Banchs. Revisiting 802.11 Rate Adaptation from Energy Consumption's Perspective. In *Proceedings of the 19th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, MSWiM '16, pages 27–34, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4502-6. DOI: 10.1145/2988287.2989149

Part III Simulation

7 Performance of RA-TPC Algorithms in 802.11

So far, we have demonstrated through numerical analysis, and validated experimentally[1], the existence of a trade-off between two competing performance figures, namely, goodput and energy efficiency. This issue arises even for a single spatial stream in absence of interference. Furthermore, we have discussed in Section 6.3.2 some ideas about the kind of mechanisms that energy-aware RA-TPC algorithms may incorporate, to leverage the behaviour that we have identified in our analysis in these so-called *mode transitions*. Summing up, the algorithms should be *conservative* during these transitions.

During that discussion, we neglected the challenge of estimating channel conditions. But in practice, any RA-TPC algorithm has imperfect channel knowledge, and therefore will adapt to changing conditions in a suboptimal way. In this chapter, we will analyse and compare the performance of several representative existing RA algorithms, which also incorporate TPC, to confirm whether the *conservativeness* in such decisions may have a positive impact on the achieved performance.

In the following, we first present the algorithms considered and describe a simple simulation scenario to test them. Our results will lead us to a more detailed discussion about *conservativeness* at mode transitions.

[1] See Appendix B.

7.1 Algorithms Considered

If we take a look at the actual operation of WiFi networks, the Minstrel[2] algorithm, which was integrated into the Linux kernel, has become the *de facto* standard due to its relatively good performance and robustness. However, Minstrel does not consider TPC and, in consequence, there is no TPC in today's WiFi deployments. Moreover, despite some promising proposals have been presented in the literature, there are very few of them implemented, although there are some ongoing efforts such as the work by the authors of *Minstrel-Piano* [Huehn and Sengul, 2012], who are pushing to release an en-

[2] A. Mcgregor and D. Smithies. Rate adaptation for 802.11 wireless networks: Minstrel, 2017. URL http://blog.cerowrt.org/papers/minstrel-sigcomm-final.pdf

hanced version of the latter for the Linux kernel with the goal of promoting it upstream.[3]

[3] https://github.com/thuehn/Minstrel-Blues

As already stated, RA is a very prolific research line in the literature, but the main corpus is dedicated to the MCS adjustment without taking into account the TXP[4]. There is some work considering TPC, but the motivation is typically the performance degradation due to network densification, and the aim is interference mitigation and not energy efficiency. Given that we are interested in assessing RA implementations with TPC support, we consider only open-source algorithms that can be tested using the NS-3 Network Simulator[5]. After a thorough analysis of the literature, we considered the following set of algorithms:

[4] See Biaz and Wu [2008] for a survey.

[5] https://www.nsnam.org

Power-controlled Auto Rate Fallback (PARF),[6] which is based on *Auto Rate Fallback* (ARF) [Kamerman and Monteban, 1997b], one of the earliest RA schemes for 802.11. ARF rate adaptation is based on the frame loss ratio. It tunes the MCS in a very straightforward and intuitive way. The procedure starts with the lowest possible MCS. Then, if either a timer expires or the number of consecutive successful transmissions reach a threshold, the MCS is increased and the timer is reset. The MCS is decreased if either the first transmission at a new rate fails or two consecutive transmissions fail. PARF builds on ARF and tries to reduce the TXP if there is no loss until a minimum threshold is reached or until transmissions start to fail. If transmission fails persist, the TXP is increased.

[6] A. Akella, G. Judd, S. Seshan, and P. Steenkiste. Self-management in chaotic wireless deployments. In *Proceedings of the 11th Annual International Conference on Mobile Computing and Networking*, MobiCom '05, pages 185–199, New York, NY, USA, 2005. ACM. ISBN 1-59593-020-5. DOI: 10.1145/1080829.1080849

Minstrel-Piano (MP)[7] is based on Minstrel [Mcgregor and Smithies, 2017]. Minstrel performs per-frame rate adaptation based on throughput. It randomly probes the MCS space and computes an exponential weighted moving average (EWMA) on the transmission probability for each rate, in order to keep a long-term history of the channel state. As the previous algorithm, MP adds TPC without interfering with the normal operation of Minstrel. It incorporates to the TPC the same concepts and techniques than Minstrel uses for the MCS adjustment, i.e., it tries to learn the impact of the TXP on the achieved throughput.

[7] T. Huehn and C. Sengul. Practical power and rate control for wifi. In *2012 21st International Conference on Computer Communications and Networks (ICCCN)*, pages 1–7, July 2012. DOI: 10.1109/ICCCN.2012.6289295

Robust Rate and Power Adaptation Algorithm (RRPAA) and *Power, Rate and Carrier-Sense Control* (PRCS),[8] which are based on *Robust Rate Adaptation Algorithm* (RRAA) [Wong et al., 2006]. RRAA consists of two functional blocks, namely, rate adaptation and collisions elimination. It performs rate adaptation based on loss ratio estimation over short windows, and reduces collisions with a RTS-based strategy. The procedure starts at the maximum MCS. The loss ratio for each window of transmissions is available for rate adjustment in the next window. There are two thresholds involved in this adjust-

[8] M. Richart, J. Visca, and J. Baliosian. Rate, power and carrier-sense threshold coordinated management for high-density ieee 802.11 networks. In *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 139–146, May 2015. DOI: 10.1109/INM.2015.7140286

ment: if the loss ratio is below both of them, the MCS is increased; if it is above, the MCS is decreased; and if it is in between, the MCS remains unchanged. RRPAA and PRCS build on this and try to use the lowest possible TXP without degrading the throughput. For this purpose, they firstly find the best MCS at the maximum TXP and, from there, they jointly adjust the MCS and TXP for each window based on a similar thresholding system. RRPAA and PRCS are very similar and only differ in implementation details.

Based on their behaviour, these algorithms can be classified into three distinct classes. First of all, MP is the most aggressive technique, given that it constantly samples the whole MCS/TXP space searching for the best possible configuration. On the opposite end, RRPAA and PRCS do not sample the whole MCS/TXP space. Instead, they are based on a windowed estimation of the loss ratio, which makes the MCS/TXP transitions much lazier. Finally, PARF falls in between, as it changes the MCS/TXP to the next available proactively if a number of transmissions are successful, but it falls back to the previous one if the new one fails. In practice, this may result in some instability during transitions.

7.2 *Simulation Scenario*

This evaluation is publicly available,[9] and builds upon the code provided by Richart et al. [2015][10]. We assessed the proposed algorithms in the toy scenario depicted in Figure 7.1. It consists of a single access point (AP) and a single mobile node connected to this AP configured with the 802.11a PHY. The mobile node at the farthest distance at which is able to communicate at the lowest possible rate (6 Mbps) and highest TXP (17 dBm), and then it moves at constant speed towards the AP. The simulation stops when the node is directly in front of the AP and it is able to communicate at the highest possible rate (54 Mbps) and lowest TXP (0 dBm). This way, we sweep through all mode transitions available.

[9] https://github.com/Enchufa2/ns-3-dev-git

[10] https://github.com/mrichart/ns-3-dev-git

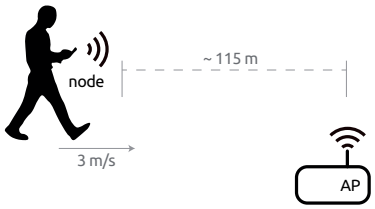


Figure 7.1: Simulation scenario.

For the whole simulation, the AP tries to constantly saturate the channel by sending full-size UDP packets to the node. Every transmission attempt is monitored, as well as every successful transmission. The first part allows us to compute the transmission time, while the latter allows us to compute the reception time (of the ACKs) and the goodput achieved.

The simulation model assembles the power model from Equation (1.1) with the parametrisation previously made (see Table 6.2) for all the devices considered in Section 6.2: HTC Legend, Linksys WRT54G, Raspberry Pi, Samsung Galaxy Note 10.1 and Soekris

net4826-48. Thus, the total energy consumed is computed for all the devices and each run using the computed transmission time, reception time and idle time. Beacons are ignored and considered as idle time.

We set up one simulation for each algorithm (PARF, MP, PRCS, RRPAA) with a fixed seed, and perform 10 independent runs for each simulation. We use boxplots to show aggregated results unless otherwise mentioned.

7.3 *Results and Discussion*

We first analyse the goodput achieved per each algorithm, which are depicted in Figure 7.2. The median of the average goodput across several runs for RRPAA is the highest, followed by PRCS, PARF and MP. PRCS and RRPAA, which are very similar mechanisms, show a higher variability across replications compared to PARF and MP, which have little dispersion.

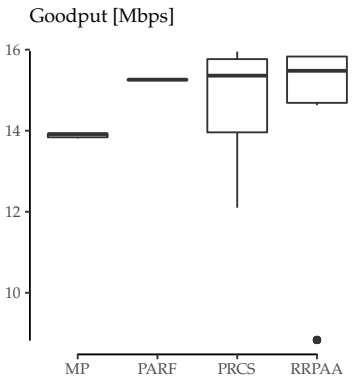


Figure 7.2: Goodput achieved per simulated algorithm.

Figure 7.3 shows the energy efficiency achieved per algorithm, computed for all the devices considered. As expected, the numerical values of the energy efficiency achieved are different across devices, but the relative performance is essentially the same, as in the previous case. Indeed, the efficiency follows the pattern seen in Figure 7.2: RRPAA results the most energy efficient in our scenario, followed by PRCS, PARF and MP. PRCS and RRPAA exhibit the same variability across replications as in the case of goodput, which is particularly notable for the most efficient devices, i.e., the HTC Legend and the Samsung Galaxy Note.

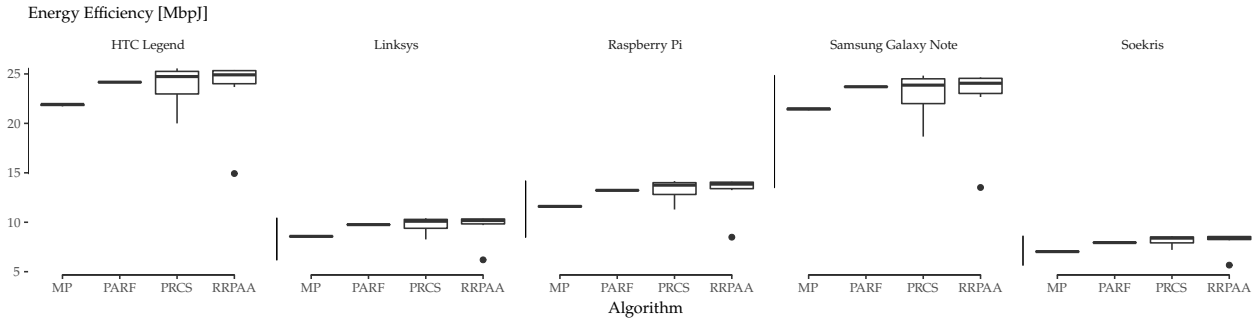


Figure 7.3: Energy efficiency achieved per simulated algorithm and device.

In order to shed some light into the reasons behind the differences in performance, Figures 7.4 show the behaviour of each algorithm throughout the simulation time for one run, showing the evolution of the MCS and TXP chosen by each algorithm, respectively. Here, we can clearly differentiate that there are two kinds of behaviour: while MP and PARF are constantly sampling other MCSs and TXPs, PRCS

and RRPAA are much more conservative in that sense, and tend to keep the same configuration for longer periods of time.

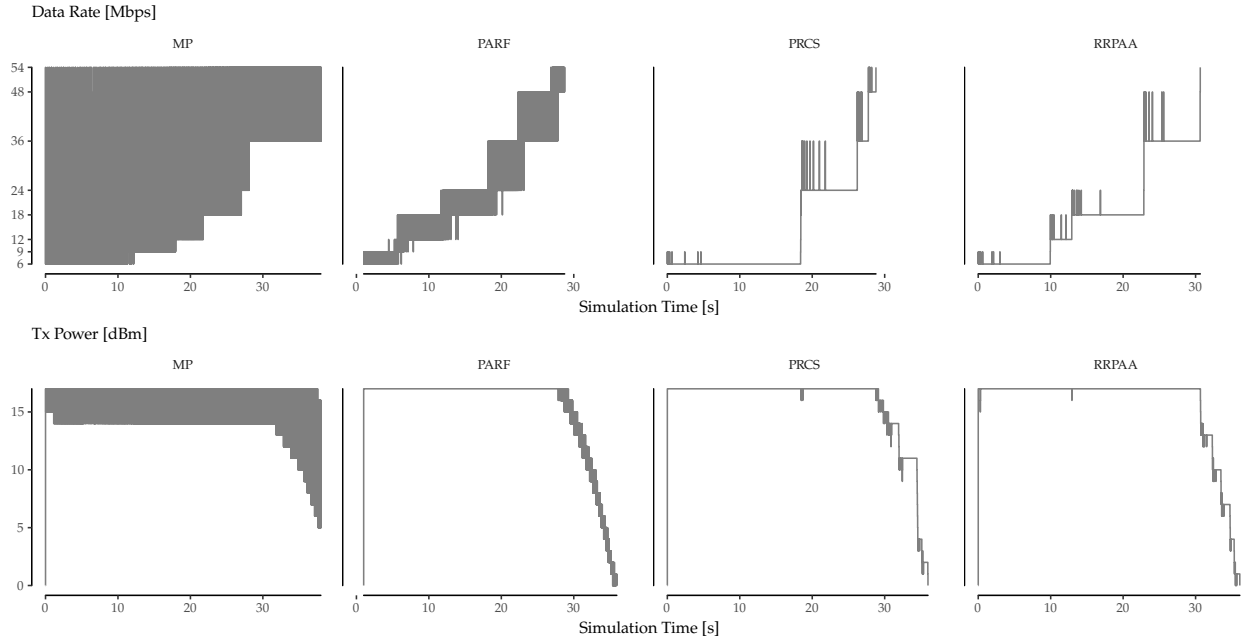


Figure 7.4: MCS (top) and TXP (bottom) evolution per algorithm for a selected run.

MP randomly explores the whole MCS/TXP space above a basic *guaranteed* value, and this is the explanation for the apparently uniformly greyed zone. Also, this aggressive approach is clearly a disadvantage in the considered toy scenario (deterministic walk, one-to-one, no obstacles), and this is why the achieved goodput in Figure 7.2 is slightly smaller than the one achieved by the others. PARF, on its part, only explores the immediately higher MCS/TXP, which leads to a higher goodput and efficiency.

On the other hand, PRCS and RRPAA sampling is much more sparse in time. As a consequence, Figures 7.4 show more differences across replications, leading to the high variability shown in Figure 7.2 compared to MP and PARF.

In terms of TXP, all the algorithms exhibit a similar *aggressiveness*, in the sense that they use a high TXP value in general. Indeed, as Figure 7.4 (bottom) shows, the TXP is the highest possible until the very end of the simulation, when the STA is very close to the AP. This is the cause for the high correlation between Figures 7.2 and 7.3.

A noteworthy characteristic of PRCS and RRPAA is that, in general, they *delay* the MCS change decision, as depicted in Figure 7.4 (top). Most of the times, they do not even use the whole space of MCS available, unlike MP and PARF. Because of this, they tend to achieve the best goodput and energy efficiency.

7.4 *Conservativeness at Mode Transitions*

Building on the concept of *conservativeness* (i.e., the tendency to select a lower MCS/TXP in the transition regions), we explore whether there is any correlation with the energy efficiency achieved by a certain algorithm and this tendency. For that purpose, we first define a proper metric.

In the first place, we define the *normalised average MCS* as the area under the curve in Figure 7.4 (top) normalised by the total simulation time and the maximum MCS:

$$\widehat{\mathrm{MCS}} = \frac{1}{\max(\mathrm{MCS}) \cdot t_{\mathrm{sim}}} \int_0^{t_{\mathrm{sim}}} \mathrm{MCS}(t)dt \tag{7.1}$$

where t_{sim} is the simulation time and $\max(\mathrm{MCS})$ is 54 Mbps in our case. The same concept can be applied to the TXP:

$$\widehat{\mathrm{TXP}} = \frac{1}{\max(\mathrm{TXP}) \cdot t_{\mathrm{sim}}} \int_0^{t_{\mathrm{sim}}} \mathrm{TXP}(t)dt \tag{7.2}$$

where $\max(\mathrm{TXP})$ is 17 dBm in our case. Both $\widehat{\mathrm{MCS}}$ and $\widehat{\mathrm{TXP}}$ are unitless scores between 0 and 1, and lower values mean a more conservative algorithm. Therefore, we can define a *Conservativeness Index* (CI) as the inverse of the product of both scores:

$$\mathrm{CI} = \frac{1}{\widehat{\mathrm{MCS}} \cdot \widehat{\mathrm{TXP}}} \tag{7.3}$$

where $\mathrm{CI} > 1$.

We computed the CI[11] for each device and run, and the final results are depicted in Figure 7.5 as the average CI across different runs vs. the median energy efficiency in Figure 7.3.

The results in Figure 7.5 show a positive non-linear relationship between the CI of an algorithm and the energy efficiency achieved for all the devices considered. MP is the algorithm with the lowest CI, which is in consonance with its aggressiveness (i.e., frequent jumps between MCS/TXP values, as shown in Figures 7.4, and the goodput

[11] It must be taken into account that the CI is not suitable for comparing *any* algorithm. For instance, in an extreme case, an "algorithm" could select 6 Mbps and 0 dBm always, resulting in a very low CI, but a very bad performance at the same time. The CI should only be used for comparing similarly performant algorithms, as it is the case in our study given the results shown in Figures 7.2 and 7.3.

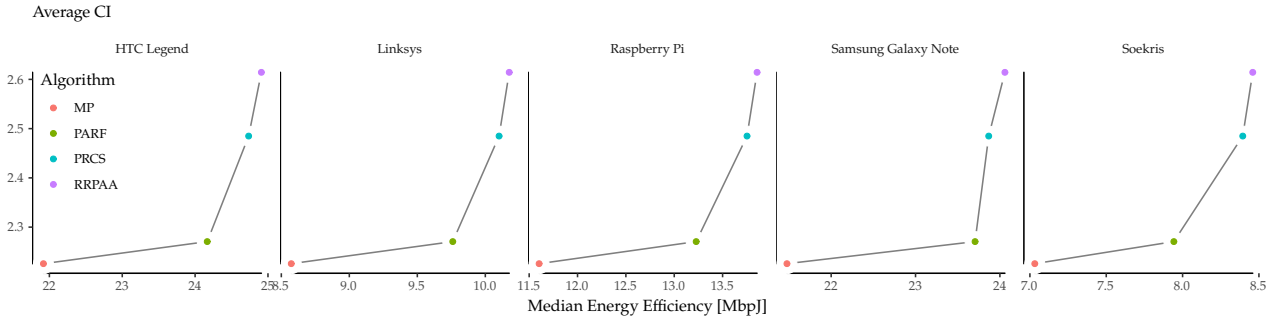


Figure 7.5: Relationship between Conservativeness Index (tendency to select lower MCS and TXP) and energy effienciency per simulated device.

achieved was also the lowest, as depicted in Figure 7.2. On the other hand, PARF, PRCS and RRPAA achieved a similar performance in terms of goodput, but the ones with the most conservative behaviour (PRCS and RRPAA, as it can be seen in Figures 7.4) also achieve both the highest CI and energy efficiency.

This result evidences that the performance gaps uncovered by Figure 6.4 under optimal conditions have also an impact in real-world RA-TPC algorithms. Therefore, we confirm that this issue must be taken into account in the design of more energy-efficient rate and transmission power control algorithms.

7.5 *Summary*

We have extended[12] our results from Chapter 6 regarding the role of *conservativeness* at mode transitions in achieving better energy efficiency in RA-TPC algorithms. We have developed a metric to compare algorithms, and we have assessed the performance of four state-of-the-art schemes through simulation. We have demonstrated that certain conservativeness can resolve the trade-off between throughput and energy efficiency optimality, thus making a difference for properly designed energy-aware algorithms.

Further research is needed to develop proper heuristics to leverage these findings. In particular, the *downwards* direction, as described in Section 6.3.2, is the most challenging, because it requires predicting the evolution of the channel state.

[12] **I. Ucar**, C. Donato, P. Serrano, A. Garcia-Saavedra, A. Azcorra, and A. Banchs. On the energy efficiency of rate and transmission power control in 802.11. *Computer Communications*, 117: 164–174, Feb. 2018a. ISSN 0140-3664. DOI: 10.1016/j.comcom.2017.07.002

8 A Novel Discrete-Event Simulation Framework

Simulation frameworks are important tools for the analysis and design of communication networks and protocols, but they can result extremely costly and/or complex (for the case of very specialised tools), or too naive and lacking proper features and support (for the case of ad-hoc tools). Our own research experience in the previous chapter has pointed us towards the need for new tools sitting between these two paradigms, supporting the efficient simulation of relatively complex scenarios at a low implementation cost.

In this chapter, we introduce a recent event-driven simulation package, `simmer`, and show its applicability in fast prototyping. `simmer` sits between the above two complexity extremes, and combines a number of features that supports, among others, versatility and repeatability. More specifically, some of the key advantages of `simmer` are as follows:

- It is based on the very popular R programming language[1], which benefits from a large community of users and contributors, but also natively supports the analysis of results via the many R statistical and visualization packages.
- The code has been peer-reviewed, and it is an official package[2], with numerous examples readily available, potentially supported by a notable user population.
- In addition to its ease of use and versatility, its code is partially optimised for speed, and therefore it can simulate relatively complex scenarios under reasonable times.

In the following, we first describe the simulation core design and its architecture. Then we provide a quick overview of `simmer` and its key features. Finally, we showcase the versatility of `simmer` to easily model a Massive Internet-of-Things (IoT) scenario where thousands of metering devices share the same channel. Here, we analyse the impact of access parameters on performance, with a particular interest in the energy required to deliver the information, which will ultimately impact the lifetime of devices running on batteries.

[1] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2017. URL https://www.R-project.org/

[2] **I. Ucar** and B. Smeets. *simmer: Discrete-Event Simulation for R*, 2018. URL https://CRAN.R-project.org/package=simmer. R package version 3.7.0

8.1 The Simulation Core Design

The core of any modern discrete-event simulator comprises two main components: an event list, ordered by time of occurrence, and an event loop that extracts and executes events. In contrast to other interpreted languages such as Python, which is compiled by default to an intermediate byte-code, R code is purely parsed and evaluated at runtime[3]. This fact makes it a particularly slow language for Discrete-Event Simulation (DES), which consists of executing complex routines (pieces of code associated to the events) inside a loop while constantly allocating and deallocating objects (in the event queue).

In fact, first attempts were made in pure R by these authors, and a minimal process-based implementation with `R6` classes[4] proved to be unfeasible in terms of performance compared to similar approaches in pure Python. For this reason, it was decided to provide a robust and fast simulation core written in C++. The R API interfaces with this C++ core by leveraging the `Rcpp` package[5], which has become one of the most popular ways of extending R packages with C or C++ code.

The following sections are devoted to describe the simulation core architecture. First, we establish the DES terminology used in the rest of the paper. Then, the architectural choices made are discussed, as well as the event queue and the *simultaneity problem*, an important topic that every DES framework has to deal with.

[3] Some effort has been made in this line with the `compiler` package, introduced in R version 2.13.0 [Luke Tierney, 2016], furthermore, a JIT-compiler was included in R version 3.4.0.

[4] W. Chang. *R6: Classes with Reference Semantics*, 2017. URL `https://CRAN.R-project.org/package=R6`. R package version 2.2.2

[5] D. Eddelbuettel and R. François. Rcpp: Seamless r and c++ integration. *Journal of Statistical Software*, 40(8):1–18, 2011. DOI: 10.18637/jss.v040.i08; and D. Eddelbuettel. *Seamless R and C++ Integration with Rcpp*. Springer-Verlag, New York, NY, USA, 2013. ISBN 978-1-4614-6867-7

8.1.1 Terminology

This document uses some DES-specific terminology, e.g., *event*, *state*, *entity*, *process* or *attribute*. Such standard terms can be easily found in any textbook about DES (refer to Banks [2005][6], for instance). There are, however, some `simmer`-specific terms, and some elements that require further explanation to understand the package architecture.

[6] J. Banks. *Discrete-event System Simulation*. Prentice-Hall International Series in Industrial and Systems Engineering. Pearson Prentice Hall, 2005. ISBN 9780131446793

Resource A passive entity, as it is commonly understood in standard DES terminology. However, `simmer` resources are conceived with queuing systems in mind, and therefore they comprise two internal self-managed parts:

- *Server* which, conceptually, represents the resource itself. It has a specified capacity and can be seized and released.
- *Queue* A priority queue of a certain size.

Manager An active entity, i.e., a process, that has the ability to adjust properties of a resource (capacity and queue size) at run-time.

Source A process responsible for creating new *arrivals* with a given interarrival time pattern and inserting them into the simulation model.

Arrival A process capable of interacting with resources or other entities of the simulation model. It may have some attributes and prioritisation values associated and, in general, a limited lifetime. Upon creation, every arrival is attached to a given *trajectory*.

Trajectory An interlinkage of *activities* constituting a recipe for arrivals attached to it, i.e., an ordered set of actions that must be executed. The simulation model is ultimately represented by a set of trajectories.

Activity The individual unit of action that allows arrivals to interact with resources and other entities, perform custom routines while spending time in the system, move back and forth through the trajectory dynamically, and much more.

8.1.2 *Architecture*

Extending an R package (or any other piece of software written in any interpreted language) with compiled code poses an important trade-off between performance and flexibility: placing too much functionality into the compiled part produces gains in performance, but degrades modelling capabilities, and vice versa. The following lines are devoted to discuss how this trade-off is resolved in `simmer`.

Figure 8.1 sketches a UML (Unified Modelling Language) description of the architecture, which constitutes a process-based design, as in many modern DES frameworks. We draw the attention now to the C++ classes (depicted in white).

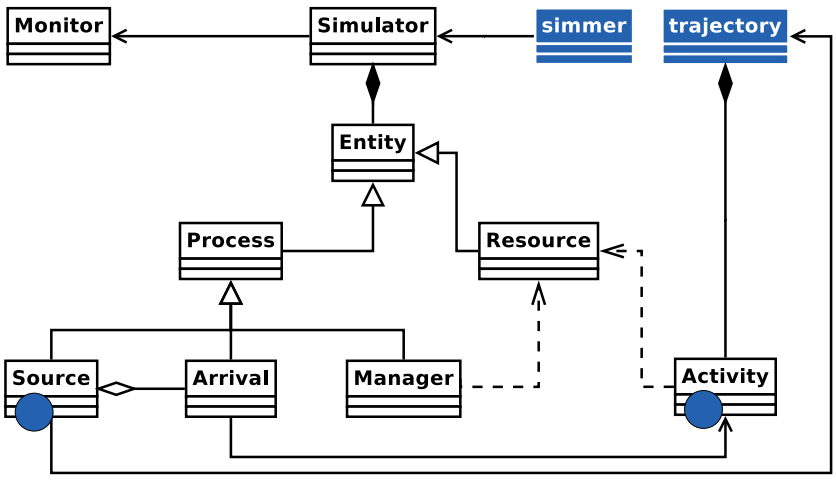


Figure 8.1: UML diagram of the simulation core architecture. Blue classes represent how R encapsulates the C++ core. Blue circles represent how C++ interfaces with R.

The first main component is the `Simulator` class. It comprises the event loop and the event queue, which will be addressed in the next section. The `Simulator` provides methods for scheduling and unscheduling events. Moreover, it is responsible for managing simulation-wide entities (e.g., resources and arrival sources) and facilities (e.g., signaling between processes and batches) through diverse C++ unordered maps:

- Maps of resources and processes (sources, arrivals and managers) by name.
- A map of pending events, which allows to unschedule a given process.
- Maps of signals subscribed by arrivals and handlers defined for different signals.
- Maps for forming batches of arrivals, named and unnamed.

This class also holds global attributes and monitoring information. Thus, monitoring counters, which are derived from the `Monitor` class, are centralised, and they register every change of state produced during the simulation time. There are five types of built-in changes of state that are recorded by calling `Monitor`'s `record_*()` methods:

- An arrival is accepted into a resource (served or enqueued). The resource notifies about the new status of its internal counters.
- An arrival leaves a resource. The resource notifies the new status of its internal counters, and the arrival notifies start, end and activity times in that particular resource.
- A resource is modified during runtime (i.e., a change in the capacity or queue size). The resource notifies the new status of its internal counters.
- An arrival modifies an attribute, one of its own or a global one. The arrival notifies the new value.
- An arrival leaves its trajectory by exhausting the activities associated (considered as *finished*) or because of another reason (*non-finished*, e.g., it is rejected from a resource). The arrival notifies global start, end and activity times.

As mentioned in the previous section, there are two types of entities: passive ones (`Resource`) and active ones (processes `Source`, `Arrival` and `Manager`). Sources create new arrivals, and the latter are the main actors of the simulation model. Managers can be used for dynamically changing the properties of a resource (capacity and queue size). All processes share a `run()` method that is invoked by the event loop each time a new event is extracted from the event list.

There is a fourth kind of process not shown in Figure 8.1, called `Task`. It is a generic process that executes a given function once, and

it is used by arrivals, resources, activities and the simulator itself to trigger dynamic actions or split up events. A `Task` is for instance used under the hood to trigger reneging or to broadcast signals after some delay.

The last main component, completely isolated from the `Simulator`, is the `Activity` class. This abstract class represents a clonable object, chainable in a double-linked list to form trajectories. Most of the activities provided by `simmer` derive from it. `Fork` is another abstract class (not depicted in Figure 8.1) which is derived from `Activity`. Any activity supporting the definition of sub-trajectories must derive from this one instead, such as `Seize`, `Branch` or `Clone`. All the activities must implement the virtual methods `print()` and `run()`.

Finally, it is worth mentioning the couple of blue circles depicted in Figure 8.1. They represent the *points of presence* of R in the C++ core, i.e., where the core interfaces back with R to execute custom user-defined code.

In summary, the C++ core is responsible for all the heavy tasks, i.e., managing the event loop, the event list, generic resources and processes, collecting all the statistics, and so on. And still, it provides enough flexibility to the user for modelling the interarrival times from R and execute any custom user-defined code through the activities.

8.1.3 The Event Queue

The event queue is the most fundamental part of any DES software. It is responsible for maintaining a list of events to be executed by the event loop in an ordered fashion by time of occurrence. This last requirement establishes the need for a data structure with a low access, search, insertion and deletion complexity. A binary tree is a well-known data structure that satisfies these properties, and it is commonly used for this purpose. Unfortunately, binary trees, or equivalent structures, cannot be efficiently implemented without pointers, and this is the main reason why pure R is very inefficient for DES.

In `simmer`, the event queue is defined as a C++ multiset, a kind of associative container implemented as a balanced tree internally. Apart from the efficiency, it was selected to support event unscheduling through iterators. Each event holds a pointer to a process, which will be retrieved and run in the event loop. Events are inserted in the event queue ordered by 1) time of occurrence and 2) priority. This secondary order criterion is devoted to solve a common issue for DES software called *the simultaneity problem*.

The Simultaneity Problem As noted by Rönngren and Liljenstam [1999] and Jha and Bagrodia [2000][7], there are many circumstances from which simultaneous events (i.e., events with the same timestamp) may arise. How they are handled by a DES framework has critical implications on reproducibility and simulation correctness.

As an example of the implications, let us consider an arrival seizing a resource at time t_{i-1}, which has `capacity=1` and `queue_size=0`. At time t_i, two simultaneous events happen: 1) the resource is released, and 2) another arrival tries to seize the resource. It is indisputable what should happen in this situation: the new arrival seizes the resource while the other continues its path. But note that if 2) is executed *before* 1), the new arrival is rejected (!). Therefore, it is obvious that release events must always be executed *before* seize events.

If we consider a dynamically managed resource (i.e., its capacity changes over time) and, instead of the event 1) in the previous example, the manager increases the capacity of the resource, we are in the very same situation. Again, it is obvious that resource managers must be executed *before* seize attempts.

A further analysis reveals that, in order to preserve correctness and prevent a simulation crash, it is necessary to break down resource releases in two parts with different priorities: the release in itself and a post-release event that tries to serve another arrival from the queue. Thus, every resource manager must be executed *after* releases and *before* post-releases. This and other issues are solved with a priority system (see Table 8.1) embedded in the event list implementation that provides a deterministic and consistent execution of simultaneous events.

[7] R. Rönngren and M. Liljenstam. On event ordering in parallel discrete event simulation. In *Proceedings of the Thirteenth Workshop on Parallel and Distributed Simulation*, PADS '99, pages 38–45, Washington, DC, USA, 1999. IEEE Computer Society. ISBN 0-7695-0155-9; and V. Jha and R. Bagrodia. Simultaneous events and lookahead in simulation protocols. *ACM Trans. Model. Comput. Simul.*, 10(3):241–267, July 2000. ISSN 1049-3301. DOI: 10.1145/361026.361032

Table 8.1: Priority system (in decreasing order) and events associated.

`Priority`	Event
`MAX`	Terminate arrivals
`RELEASE`	Resource release
`MANAGER`	Manager action (e.g., resource capacity change)
`RELEASE_POST`	Resource post-release (i.e., serve from the queue)
`...`	General activities
`MIN`	Other tasks (e.g., new arrivals, timers...)

8.2 *A Brief Introduction to `simmer`*

Note that `simmer` does not aim at substituting NS-3 or OMNeT++, which are the *de facto* standards for open-source network simulations. Instead, `simmer` is designed as a general-purpose DES framework

with a human-friendly syntax, and a very gentle learning curve. It can be used to complement other field-specific simulators as a rapid prototyping tool that enable insightful analysis of different designs. As we will illustrate in the next section, with `simmer` it is simple to simulate relatively complex scenarios, with the added benefit of the availability of many convenient data analysis and representation packages, thanks to the use of R.

The R application programming interface (API) exposed by `simmer` revolves around the concept of *trajectory*, which defines the "path" in the simulation for entities of the same type. A trajectory is a recipe for the arrivals attached to it, an ordered set of actions (or *verbs*) chained together with the pipe operator[8] (`%>%`, whose behaviour is similar to the command-line pipe). The following example illustrates a basic `simmer` workflow, modeling the classical case of customers being attended by a single clerk with infinite waiting space in a few lines of code:

[8] S. M. Bache and H. Wickham. *magrittr: A Forward-Pipe Operator for R*, 2014. URL https://CRAN.R-project.org/package=magrittr. R package version 1.5

```
library(simmer)

cust <- trajectory("customer") %>%
  seize("clerk", amount=1) %>%
  timeout(function() rexp(1, 2)) %>%
  release("clerk", amount=1)

env <- simmer("bank") %>%
  add_resource("clerk", capacity=1, queue_size=Inf) %>%
  add_generator("cust", cust, function() rexp(1, 1)) %>%
  run(until=1000)

arrivals  <- get_mon_arrivals(env)
resources <- get_mon_resources(env)
```

Given that both the time at the clerk and between customers are exponential random variables, and the infinite queue length, this example corresponds, in Kendall's notation, to an M/M/1 queue. It serves to illustrate the two main elements of `simmer`: the `trajectory` object and the `simmer` environment (or *simulation environment*).

The `customer` trajectory defines the behaviour of a generic customer: seize a clerk, spend some time, and release it. The `env` simulation environment is then defined as one clerk with infinite queue size and a generator of customers, each one following the trajectory defined above. Based on this syntax, the flexibility is provided through a rich set of activities[9] that can be appended to trajectories, which support: changing arrivals' properties (attributes, priority, batches), different interactions with the resources (select, seize, release, change

[9] See https://r-simmer.org/reference/.

their properties), and the generators (activate, deactivate, change their properties), and even the definition of branches (simple, depending on a condition, or parallel) and loops. Finally, some support to asynchronous programming is also provided (subscription to signals and registration of handlers).

Not only `simmer` provides a powerful yet simple syntax, but it is also *fast*, for example, faster than equivalent frameworks such as SimPy[10] and SimJulia[11] for the Python and Julia languages respectively[12]. Furthermore, and perhaps more importantly, `simmer` implements automatic monitoring capabilities: every event is accounted for by default, both for arrivals (starting and ending times, activity time, ending condition, resources traversed) and resources (server and queue status), and all this information can be easily retrieved in standard R data frames for further processing of results (last two lines of the *clerk* example).

[10] Team SimPy. *SimPy: Discrete-Event Simulation for Python*, 2017. URL `https://simpy.readthedocs.io/en/latest/`. Python package version 3.0.9

[11] B. Lauwens. *SimJulia.jl: Combined Continuous-Time / Discrete-Event Process Oriented Simulation Framework Written in Julia*, 2017. URL `https://github.com/BenLauwens/SimJulia.jl`. Julia package version 0.3.14

[12] See Appendix C for a complete performance evaluation.

8.3 *Use case: Energy Efficiency for Massive IoT*

We consider the case of a massive Internet-of-Things (mIoT) scenario, a use case for Long Term Evolution (LTE) and next-generation 5G networks, as defined by the 3GPP[13]. As Figure 8.2 (left) illustrates, we consider a single LTE macrocell in a dense urban area. The buildings in the cell area are populated with N smart meters (for electricity, gas, and water), and each meter operates independently as a Narrowband IoT (NB-IoT) device.

[13] C. Hoymann, D. Astely, M. Stattin, G. Wikstrom, J. F. Cheng, A. Hoglund, M. Frenne, R. Blasco, J. Huschke, and F. Gunnarsson. Lte release 14 outlook. *IEEE Communications Magazine*, 54(6): 44–49, June 2016. ISSN 0163-6804. DOI: 10.1109/MCOM.2016.7497765

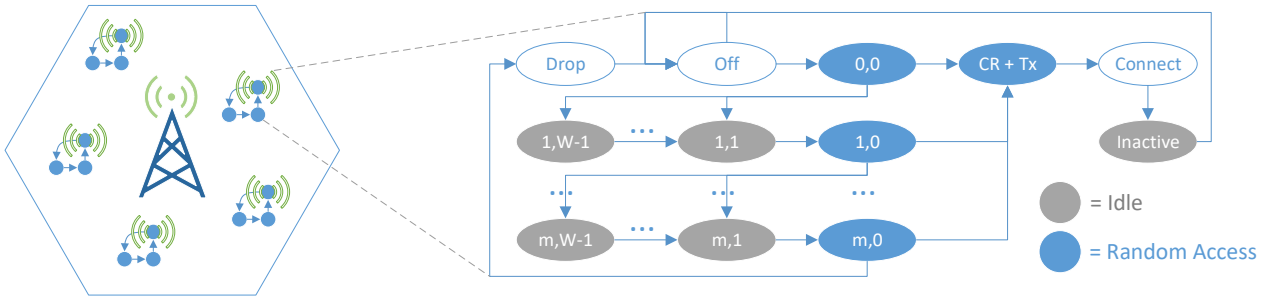


Figure 8.2: Description of the simulation scenario.

The devices' behaviour is modeled following the diagram depicted in Figure 8.2 (right), which is a simplified version of the Markov chain model developed in Andres-Maldonado et al. [2017, Figure 5][14]. A device may stay in `RRC Idle` ('Off'), and awakes with some periodicity to upload its reading. This communication phase encompasses a contention-based random access (RA) procedure, with a backoff time randomly chosen between $(0, W)$ time slots, and up to m retransmissions. If the connection request fails, the reading is dropped, and the device returns to the 'Off' state. If the connection is

[14] P. Andres-Maldonado, P. Ameigeiras, J. Prados-Garzon, J. J. Ramos-Munoz, and J. M. Lopez-Soler. Optimized lte data transmission procedures for iot: Device side energy consumption analysis. In *2017 IEEE International Conference on Communications Workshops (ICC Workshops)*, pages 540–545, May 2017. DOI: 10.1109/ICCW.2017.7962714

successful, we assume that the device implements the Control Plane Cellular IoT (CP) optimization (see Andres-Maldonado et al. [2017]), so that the data is transmitted over the `RRC Connection` request phase using the Non Access Stratum (NAS) level. Then, the device has to wait ('Inactive') until the connection is released, and eventually returns to the 'Off' state.

The goal of this use case is to study the effect of synchronization across IoT devices (for instance, due to a power outage) in the energy consumption. As in Cheng et al. [2012][15], we assume that a device provides its readings as often as every hour, and the cases of $N = \{5, 10, 30\} \cdot 10^3$ devices in one cell are considered. In order to study different levels of synchronization, each node implements an additional backoff window prior to the RA procedure. Furthermore, we selected $m = 9$ and $W = 20$; the rest of the parameters (power consumption, timings, message sizes...) can be found in Andres-Maldonado et al. [2017, Table I].

[15] R. G. Cheng, C. H. Wei, S. L. Tsao, and F. C. Ren. Rach collision probability for machine-type communications. In *2012 IEEE 75th Vehicular Technology Conference (VTC Spring)*, pages 1–5, May 2012. DOI: 10.1109/VETECS.2012.6240129

Implementation Details This scenario requires a single `meter` trajectory implementing the logic of each IoT device in an infinite loop, and N workers are attached to it at $t = 0$.

```
# IoT device logic
meter <- trajectory() %>%
  trap("reading") %>%
  # sleep
  set_attribute("P", 0) %>%
  wait() %>%
  timeout(function() round(runif(1, 0, param[["backoff"]]), 3)) %>%
  # ra start
  simmer::select(preambles, policy="random") %>%
  seize_selected(
    continue=c(TRUE, TRUE),
    # ra & tx
    post.seize=trajectory() %>%
      set_attribute("P", Pra) %>%
      timeout(Tra) %>%
      release_selected() %>%
      set_attribute("P", Ptx) %>%
      timeout(Ttx),
    # ra & backoff & retry
    reject=trajectory() %>%
      set_attribute("P", Pra) %>%
      timeout(Tra) %>%
      set_attribute("P", Pi) %>%
```

```
    timeout(function() sample(1:20, 1) * 1e-3) %>%
    rollback(6, times=m)
) %>%
rollback(5, times=Inf)
```

Each device registers itself for a given signal ("reading"), and waits in sleep mode until a new reading is requested, which is triggered by a secondary trajectory (`trigger`).

```
# trigger a reading for all the meters every tx_period
trigger <- trajectory() %>%
  timeout(tx_period) %>%
  send("reading") %>%
  rollback(2, times=Inf)
```

As soon as a new reading is signalled, the RA procedure starts by randomly selecting one of the 54 preambles available, which are defined as resources. The process of seizing a preamble encompasses two sub-trajectories:

- If there are no collisions, the preamble is successfully seized, and the `post.seize` sub-trajectory is executed, which transmits a reading.
- If there is collision, rejection occurs, and the `reject` sub-trajectory is executed, which performs the RA backoff (for a random number of slots), and restarts the RA procedure (for a maximum of m retries).

Both sub-trajectories set the appropriate power levels P for the appropriate amount of time. In this case, these power levels throughout the simulation time are retrieved with the `get_mon_attributes()` method. The energy is concisely computed and represented using the packages `dplyr`[16] and `ggplot`[17].

```
## Warning: Detecting old grouped_df format,
## replacing 'vars' attribute by 'groups'
```

Figure 8.3 shows the results for one simulated day. It depicts the energy consumed per reading considering a uniform backoff window between 0 and 5 (*highly synchronised*), 10, 30, and 60 seconds (*non-synchronised*). As the number of devices and the level of synchronization grow, the random-access opportunities (RAOs) per second grow as well producing more and more collisions. These collisions cause retries, and a noticeable impact in the energy consumption (up to 12% more energy per reading). Therefore, this use case shows the paramount importance of randomizing node activation in mIoT scenarios in order to avoid RAO peaks and a premature battery drain.

[16] H. Wickham and R. Francois. *dplyr: A Grammar of Data Manipulation*, 2017. URL `https://CRAN.R-project.org/package=dplyr`. R package version 0.7.4

[17] H. Wickham and W. Chang. *ggplot2: Create Elegant Data Visualisations Using the Grammar of Graphics*, 2016. URL `https://CRAN.R-project.org/package=ggplot2`. R package version 2.2.1

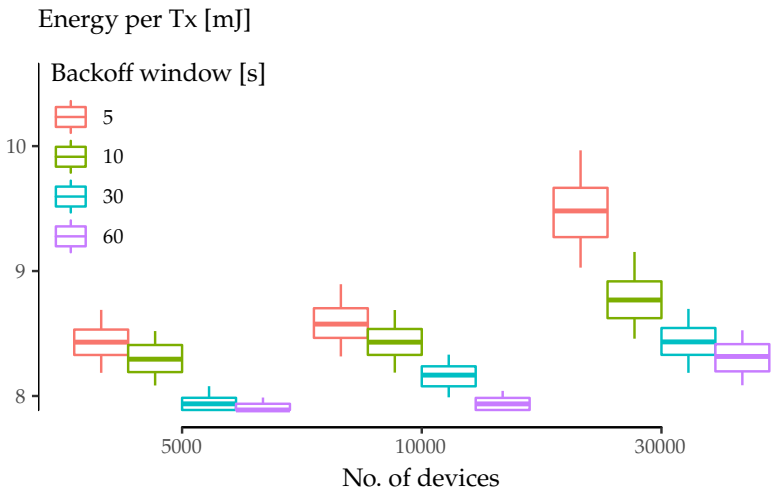


Figure 8.3: Energy consumption per transmission attempt for different traffic models and number of devices.

THE SIMULATION was run with a machine equipped with an Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz x4 (32 cores), and 64 GB of RAM, Debian GNU/Linux 8, R 3.3.2, and `simmer` 3.6. Table summarises the main simulation statistics for this scenario. These numbers attest that `simmer` can be used to simulate relatively complex scenarios with very few lines of code[18].

	Features
Simulation time (s)	150
No. of parallel scenarios	12
Max. events, 1 scenario	28364172
Total no. of events	98952165
Implementation lines	42
Analysis + plotting lines	14

Table 8.2: Overview of simulation features.

[18] The full code is available online at `https://r-simmer.org/articles/simmer-aa-5g#energy-efficiency-for-massive-iot`.

8.4 Summary

The `simmer` package presented in this chapter[19] brings a generic yet powerful process-oriented Discrete-Event Simulation framework to R. `simmer` combines a robust and fast simulation core written in C++ with a rich and flexible R API. The main modelling component is the *activity*. Activities are chained together with the pipe operator into *trajectories*, which are common paths for processes of the same type. `simmer` provides a broad set of activities, and allows the user to extend their capabilities with custom R functions.

Monitoring is automatically performed by the underlying simulation core, thereby enabling the user to focus on problem modelling. `simmer` enables simple replication and parallelisation with standard R tools. Data can be extracted into R data frames from a single simulation environment or a list of environments, each of which is marked as a different replication for further analysis.

We have demonstrated[20] the usability and suitability of `simmer` for fast prototyping of a 5G-inspired scenario. The code developed highlights some of the characteristics that make `simmer` attractive for researchers and practitioners in communications research.

[19] **I. Ucar**, B. Smeets, and A. Azcorra. simmer: Discrete-Event Simulation for R. *Journal of Statistical Software*, (accepted for publication), 2018d. ISSN 1548-7660. URL `https://arxiv.org/abs/1705.09746`

[20] **I. Ucar**, J. A. Hernández, P. Serrano, and A. Azcorra. Design and Analysis of 5G Scenarios with simmer: An R Package for Fast DES Prototyping. *IEEE Communications Magazine*, 56(11):145–151, Nov. 2018b. ISSN 0163-6804. DOI: 10.1109/MCOM.2018.1700960

- A novel and intuitive trajectory-based approach that simplifies the simulation of large networks of queues, including those with feedback.
- Flexible resources, with dynamic capacity and queue size, priority queueing and preemption.
- Flexible generators of arrivals that can draw interarrival times from any theoretical or empirical distribution via a function call.
- Asynchronous programming features and monitoring capabilities, which helps the researcher focus into the model design.

It is likewise remarkable the ease with which multiple scenarios, with different parameters, can be simulated concurrently thanks to base R functions. Thus, exploring a large number of combinations of parameter values is not only straightforward, but also as fast as the slowest thread given enough number of CPU cores available.

9 *Conclusions and Future Work*

This thesis has been devoted to the study of the energy efficiency of mobile user devices at multiple layers by means of experimentation, mathematical modelling and simulation. Although the latter two methods are important tools for science in general, and for this field in particular, we must emphasise the complexity and paramount importance of experimentation.

In a preeminent experimental part that has served as a basis for the rest of the thesis, we have assembled a comprehensive energy measurement framework, and a robust methodology, which is capable of measuring a wide range of wireless devices, as well as individual components, with high accuracy and precision. A whole-device parametrisation has been presented and validated against previous results from Serrano et al. [2015]. Similarly, a precise characterisation of a commercial off-the-shelf wireless card has been used to produce several contributions and insights throughout this thesis.

In connection with the aforementioned methodology, measurement handling has been systematised into `errors`[1], a lightweight R package that associates standard uncertainty metadata to numeric vectors, matrices and arrays. The resulting data type automatically handles propagation of uncertainty through a first-order Taylor series method, and provides a formally sound representation of measurements. Using this package makes the process of computing indirect measurements easier and less error-prone.

[1] **I. Ucar**, E. Pebesma, and A. Azcorra. Measurement Errors in R. *The R Journal*, 10(2):549–557, 2018c. ISSN 2073-4859. doi: 10.32614/RJ-2018-075

Building on this measurement platform, we have delved into the energy consumption of the kernel cross-factor, an energy toll produced by frame processing within the devices' network stack. Our whole-device measurements on a laptop computer have provided several fundamental insights[2] on this matter. Firstly, we have identified the CPU as the leading cause of the energy consumption, discarding the RAM memory as a significant component for this kind of device. Secondly, we have demonstrated that the CPU's C-state

[2] **I. Ucar**, A. Azcorra, and A. Banchs. Deseeding Energy Consumption of Network Stacks. In *6th Annual International IMDEA Networks Workshop*, June 2014. (invited poster); and **I. Ucar** and A. Azcorra. Deseeding energy consumption of network stacks. In *IEEE 1st International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI)*, pages 7–16, Sept. 2015. ISBN 978-1-4673-8167-3. doi: 10.1109/RTSI.2015.7325085

management system plays a central role in such consumption behaviour. In fact, the characterisation of the cross-factor is much more complex that the previous work showed. Specifically, the consumption depends on the C-state residence times, which in turn depend on the wake-up rate produced by software and hardware interrupts.

As a consequence, the cross-factor's linear behaviour disappears for CPUs with more than a single C-state, which is the case for many devices (laptop computers, smartphones, tablets...). Therefore, the energy model by Serrano et al. [2015] cannot be applied to such devices. The energy consumption cannot be explained as increments on top of a baseline power, but as savings from a maximum consumption in a saturated state. This fact poses doubts on the applicability of such energy model on devices with sophisticated CPUs, since the savings achieved and the level of non-linearity strongly depend on the device's general state (i.e., on how the CPU is stressed: number of applications running and type of load). Nevertheless, some practices and recommendations related to packet batching still hold, because longer loads with fewer interruptions lead to longer inactivity periods that can be leveraged by deeper sleep states.

Moreover, the methodology to perform energy breakdowns by dropping packets inside the transmission chain is no longer valid for such CPUs, because this method generates a varying rate of interrupts. Therefore, further research is needed to produce novel measurement methodologies to fully understand the key role of the C-state subsystem in the energy consumption of protocol stacks.

We then turned our attention to lower levels of the communication stack to revisit the behaviour of idle 802.11 interfaces. We have studied a commercial wireless card to understand the timing constraints that its architecture poses on state changes (i.e., from idle to sleep state and back to normal operation). This characterisation sets fundamental limits for any energy-efficiency mechanism based on powering down the wireless interface, and serves as a basis for the development of practical algorithms.

Based on this, we have analysed and unveiled the practical challenges of the micro-sleep opportunities that are available in current WLANs. A comprehensive discussion demarcates all the conditions that must be met to leverage packet overhearing as a source of energy savings without breaking the 802.11 standard. Furthermore, these conditions have been fine-tuned based on practical issues (e.g., capture effect) not previously considered in prior work.

Building on this knowledge, we have proposed[3] and patented[4] μNap a standard-compliant and incrementally-deployable energy-saving scheme that is orthogonal to existing standard PS mecha-

[3] A. Azcorra, **I. Ucar**, A. Banchs, F. Gringoli, and P. Serrano. μNap: Practical micro-sleeps for 802.11 WLANs. *Computer Communications*, 110:175–186, Sept. 2017. ISSN 0140-3664. DOI: 10.1016/j.comcom.2017.06.008

[4] A. Azcorra, **I. Ucar**, A. Banchs, F. Gringoli, and P. Serrano. Energy-saving method based on micro-shutdowns for a wireless device in a telecommunications network. WO/2018/015601, Jan. 2018. URL http://www.google.com/patents/WO2018015601

nisms. 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 through trace-based simulation 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.

At a higher level of the communication stack, we have revisited 802.11 rate adaptation (RA) and transmission power control (TPC) from the energy standpoint. Previous studies pointed out that MIMO rate adaptation under 802.11n poses a trade-off between throughput and energy efficiency maximisation. We have in turn demonstrated with an analytical model that, even for single spatial streams without interfering traffic, these are competing objectives.

Our findings[5] show that RA-TPC techniques may incur inefficiencies at mode transitions, which suggests that small goodput degradations may lead to energy efficiency gains. Several heuristics have been provided and discussed to manage those transitions in an energy-efficient manner. We have shown that transitions leading to a lower rate and transmission power are particularly challenging, as they imply predicting channel quality drops to trigger early transitions.

We have extended[6] these results by simulating several state-of-the-art RA-TPC algorithms and assessing their performance at mode transitions. The metric developed for such comparison confirms that certain *conservativeness* can resolve the trade-off between throughput and energy efficiency optimality, thus making a difference for properly designed energy-aware algorithms. Future work may delve into the proper heuristics to leverage these findings to develop novel energy-aware RA-TPC algorithms.

Finally, our research experience on simulations with highly complex network simulators pointed us towards the need for new less-specialised tools enabling easier and faster prototyping. As a result, we have developed `simmer`[7], an easy-to-use yet powerful process-oriented and trajectory-based discrete-event simulation framework for R. This package combines a flexible API and a robust and fast simulation core written in C++ with automatic monitoring capabilities, which enables the user to focus on the system model.

[5] **I. Ucar**, C. Donato, P. Serrano, A. Garcia-Saavedra, A. Azcorra, and A. Banchs. Revisiting 802.11 Rate Adaptation from Energy Consumption's Perspective. In *Proceedings of the 19th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, MSWiM '16, pages 27–34, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4502-6. DOI: 10.1145/2988287.2989149

[6] **I. Ucar**, C. Donato, P. Serrano, A. Garcia-Saavedra, A. Azcorra, and A. Banchs. On the energy efficiency of rate and transmission power control in 802.11. *Computer Communications*, 117: 164–174, Feb. 2018a. ISSN 0140-3664. DOI: 10.1016/j.comcom.2017.07.002

[7] **I. Ucar**, B. Smeets, and A. Azcorra. simmer: Discrete-Event Simulation for R. *Journal of Statistical Software*, (accepted for publication), 2018d. ISSN 1548-7660. URL `https://arxiv.org/abs/1705.09746`

We have demonstrated[8] the usability and suitability of `simmer` for fast prototyping of a 5G-inspired scenario, which consists of the energy modelling of thousands of IoT metering devices connected to an LTE macrocell. The model developed highlights in just a few lines of code the main characteristics that make `simmer` attractive for researchers and practitioners in communications research.

[8] **I. Ucar**, J. A. Hernández, P. Serrano, and A. Azcorra. Design and Analysis of 5G Scenarios with simmer: An R Package for Fast DES Prototyping. *IEEE Communications Magazine*, 56(11):145–151, Nov. 2018b. ISSN 0163-6804. DOI: 10.1109/MCOM.2018.1700960

FUTURE WORK may include the application of the methodologies developed and lessons learned in this thesis to a wider range of battery-powered devices. Particularly, the Internet-of-Things has been receiving an increased attention, and the market of wearable devices as well as sensor networks for smart cities and smart homes is growing fast. All these devices are aimed at very specific tasks, and thus they share simplicity in terms of hardware and software architecture. As a result, the linear model can be applied to describe the energy behaviour of such devices, and the measurement methodologies developed in this work are well-suited for a complete characterisation.

Within the scope of more sophisticated devices such as smartphones or laptop computers, new energy models specifically focused on modern CPUs are needed, which should be parametrised in terms of the workload. But we have shown that the total workload, understood as pure CPU cycles, is not as important as the type of workload, namely, its segmentation over time. Future lines of research may include working on CPU governor algorithms to get the best of each idle period, as well as developing buffered system calls and mechanisms to minimise the amount of interrupts per bit.

Regarding 802.11, this thesis points towards new directions to evaluate existing RA-TPC schemes in terms of energy efficiency, as well as to develop novel energy-aware RA-TPC algorithms. Furthermore, the constant evolution of the standard requires a sustained effort to fine-tune the energy consumption of a growing set of features.

But beyond 802.11, this work can be further developed and applied to other wireless access technologies such as LTE, which is in turn evolving to support the fifth-generation of mobile networks. The centralised nature of these networks poses new challenges, but also opens up new opportunities: savings may not only be achieved in the protocol itself, but also in the scheduling algorithms that are needed to share the radio resources.

Finally, although it has been assumed based on prior work that the energy consumption of 3G and 4G interfaces is higher than the consumption of 802.11 interfaces, this may not be the case in 5G. As the efficiency gap closes, further research is needed in order to achieve an optimal use, also in terms of energy, of the available technologies.

A Measurement Circuitry Schematics

The energy measurement framework presented in Chapter 3 requires custom circuitry to bypass the line that connects the power source to the DUT. The purpose is to extract and adapt the signals of interest (voltage and current) and feed them into the DAQ card within the proper input limits.

The design was conducted in our university's Technical Office in Electronics, which built and calibrated several prototypes based on two schematics for different input ranges:

- A design for 12.5-20 V and 0-5 A, depicted in Figure A.1.
- A design for 0-5 V and 0-2 A, depicted in Figure A.2.

In the schematics, `CLEMA 8` is the connector for the power source. It is designed to be powered with a Keithley 2304A DC Power Supply, but it can be attached to other power sources as well. Apart from the `SOURCE+` and `SOURCE-` terminals, there are sensing terminals and measurement terminals (`DVM`) that this Keithley power supply uses to stabilise the output voltage.

The DUT is powered through `CLEMA 4`, and the DAQ card is attached to `CLEMA 6`, where two voltage followers feed the measurement terminals. All cables from these connectors (to the DUT and DAQ respectively) must be shielded and as short as possible in order to minimise losses.

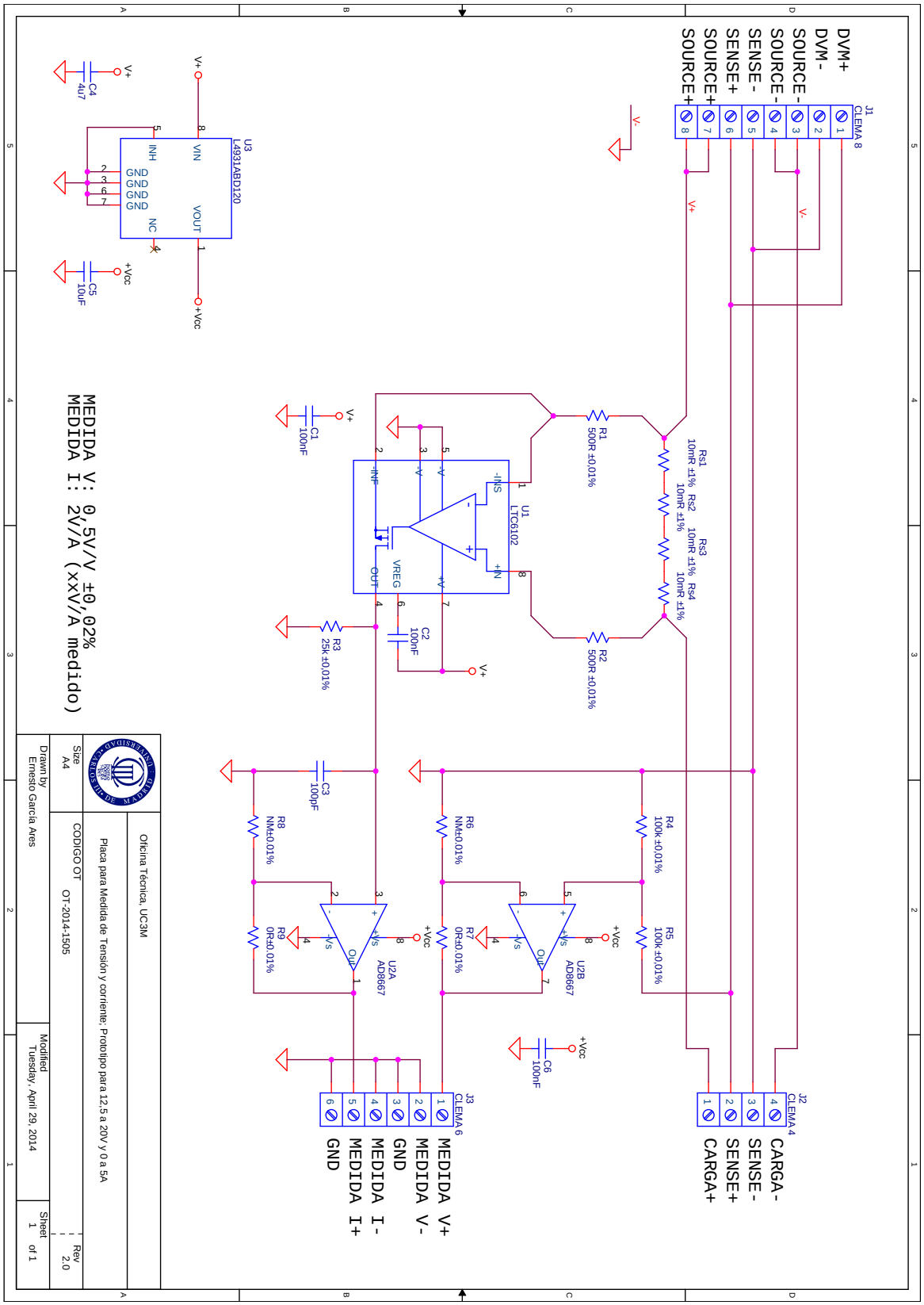


Figure A.1: 12,5-20 V and 0-5 A (100 W max.) prototype.

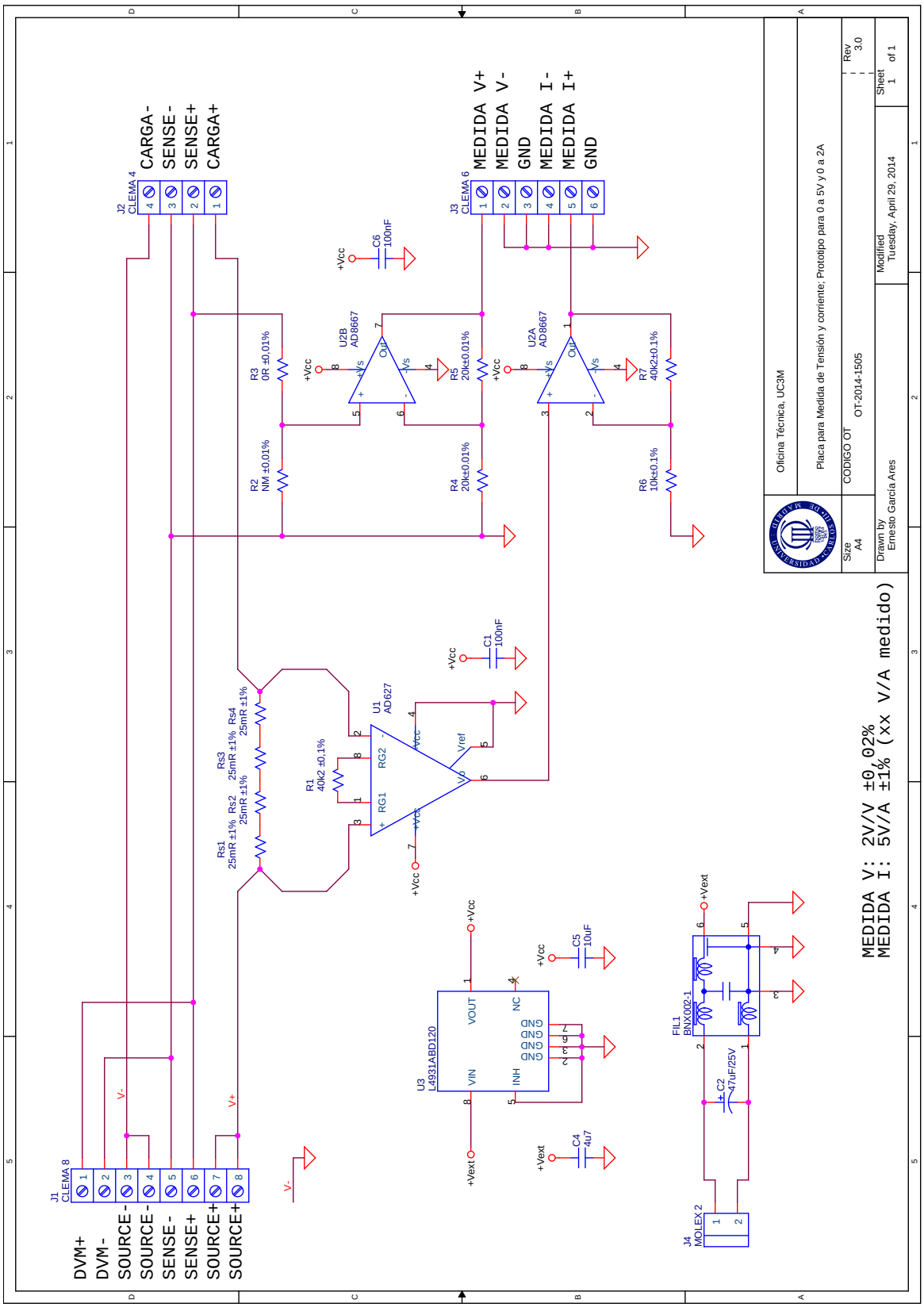


Figure A.2: 0-5 V and 0-2 A (10 W max.) prototype.

B Experimental Validation of RA-TPC Inefficiencies

This appendix is devoted to experimentally validate the results from the numerical analysis developed in Chapter 6. To this aim, we describe our experimental setup and validation procedure, first specifying the methodology and then the results achieved.

B.1 Experimental Setup

We deployed the testbed illustrated in Figure B.1, which is a variation of the one depicted in Section 3.4, Figure 3.7. It consists of a station (STA) transmitting evenly-spaced maximum-sized UDP packets to an access point (AP), an x86-based Alix6f2 board with a Atheros AR9220 wireless card, running kernel version 3.16.7 and the `ath9k` driver. The STA is a desktop PC with a Mini PCI Express Qualcomm Atheros QCA9880 wireless network adapter, running Fedora Linux 23 with kernel version 4.2.5 and the `ath10k` driver[1]. We also installed at the STA a Mini PCI Qualcomm Atheros AR9220 wireless network adapter to monitor the wireless channel.

[1] Following the discussion on Section 6.3.1 the device's cross-factor is not involved in the trade-off, thus we will expect to reproduce it by measuring the wireless interface alone.

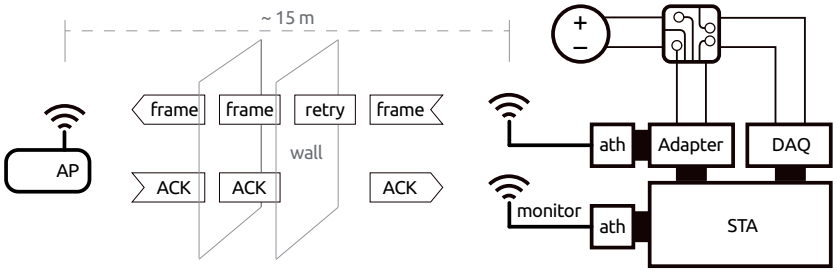


Figure B.1: Experimental setup.

As Figure B.1 illustrates, the STA is located in an office space and the AP is placed in a laboratory 15 m away, and transmitted frames have to transverse two thin brick walls. The wireless card uses only one antenna and a practically-empty channel in the 5-GHz band. Throughout the experiments, the STA is constantly backlogged with data to send to the AP, and measures the throughput obtained by counting the number of acknowledgements (ACKs) received.

B.2 *Methodology and Results*

In order to validate our results, our aim is to replicate the qualitative behaviour of Figure 6.4, in which there are energy efficiency "drops" as the optimal goodput increases. However, in our experimental setting, channel conditions are not controllable, which introduces a notable variability in the results as it affects both the x-axis (goodput) and the y-axis (energy efficiency). To reduce the impact of this variability, we decided to change the variable in the x-axis from the optimal goodput to the transmission power —a variable that can be directly configured in the wireless card—. In this way, the qualitative behaviour to replicate is the one illustrated in Figure B.2, where we can still identify the performance "drops" causing the loss in energy efficiency.

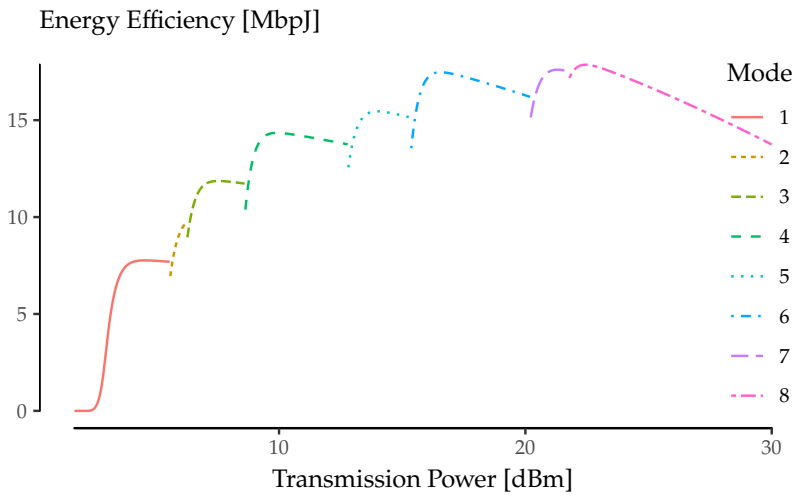


Figure B.2: Energy Efficiency vs. Transmission Power under fixed channel conditions for the Raspberry Pi case.

Building on Figure B.2, we perform a sweep through all available combinations of MCS (see Table 6.1) and TXP.[2] Furthermore, we also tested two different configurations of the AP's TXP at different times of the day, to confirm that this qualitative behaviour is still present under different channel conditions. For each configuration, we performed 2-second experiments in which we measure the total bytes successfully sent and the energy consumed by the QCA9880 card with sub-microsecond precision, and we compute the energy efficiency achieved for each experiment.

The results are shown in Figure B.3. Each graph corresponds to a different TXP value configured at the AP, and depicts a single run (note that we performed several runs throughout the day and found no major qualitative differences across them). Each line type represents the STA's mode that achieved the highest goodput for each TXP interval, therefore in some cases low modes do not appear because

[2] The model explores a range between 0 and 30 dBm to get the big picture, but this particular wireless card only allows us to sweep from 0 to 20 dBm.

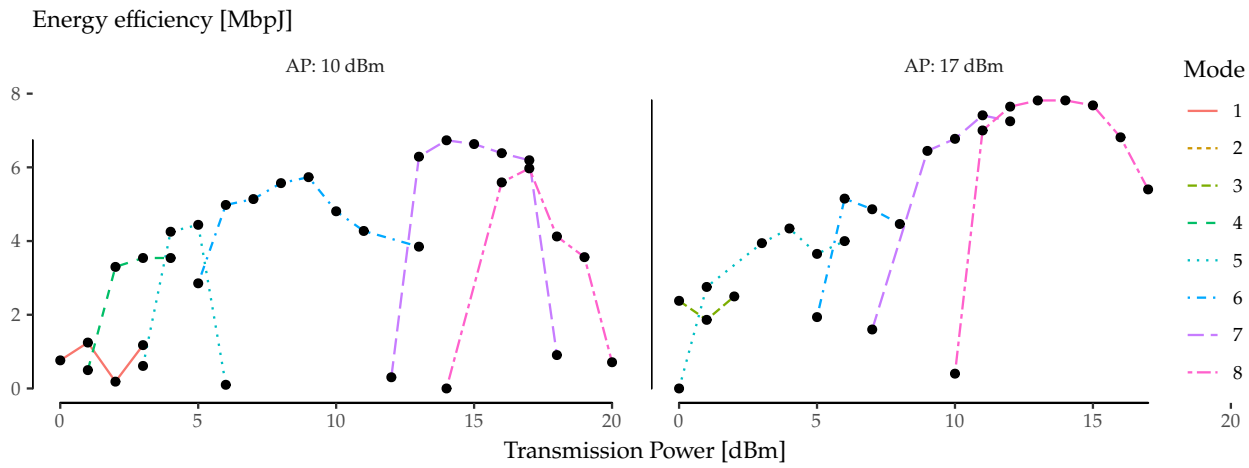


Figure B.3: Experimental study of Figure B.2 for two AP configurations.

a higher mode achieved a higher goodput. Despite the inherent experimental difficulties, namely, the low granularity of 1-dBm steps and the random variability of the channel, the experimental results validate the analytical ones, as the qualitative behaviour of both graphs follows the one illustrated in Figure B.2. In particular, the performance "drops" of each dominant mode can be clearly observed (especially for the 36, 48 and 54 Mbps MCSs) despite the variability in the results.

C Performance Evaluation of `simmer`

This appendix investigates the performance of `simmer` with the aim of assessing its usability as a general-purpose DES framework. A first section is devoted to measuring the simulation time of a simple model relative to `SimPy`[1] and `SimJulia`[2]. The reader may find interesting to compare the expressiveness of each framework. Last but not least, the final section explores the cost of calling R from C++, revealing the existent trade-off, inherent to the design of this package, between performance and model complexity.

All the subsequent tests were performed under Fedora Linux 25 running on an Intel Core2 Quad CPU Q8400, with R 3.3.3, Python 2.7.13, `SimPy` 3.0.9, Julia 0.5.1 and `SimJulia` 0.3.14 from the default repositories. Absolute execution times presented here are specific to this platform and configuration, and thus they should not be taken as representative for any other system. Instead, the relative performance should be approximately constant across different systems.

[1] Team SimPy. *SimPy: Discrete-Event Simulation for Python*, 2017. URL `https://simpy.readthedocs.io/en/latest/`. Python package version 3.0.9

[2] B. Lauwens. *SimJulia.jl: Combined Continuous-Time / Discrete-Event Process Oriented Simulation Framework Written in Julia*, 2017. URL `https://github.com/BenLauwens/SimJulia.jl`. Julia package version 0.3.14

C.1 Comparison with Similar Frameworks

A significant effort has been put into the design of `simmer` in order to make it performant enough to run general and relatively large simulation models in a reasonable amount of time. In this regard, a relevant comparison can be made against other general-purpose DES frameworks such as `SimPy` and `SimJulia`. To this effect, we simulate a simple M/M/1 queueing system as follows:

```
library("simmer")

test_mm1_simmer <- function(n, m, mon=FALSE) {
  mm1 <- trajectory() %>%
    seize("server", 1) %>%
    timeout(function() rexp(1, 1.1)) %>%
    release("server", 1)
```

```
  env <- simmer() %>%
    add_resource("server", 1, mon=mon) %>%
    add_generator("customer", mm1, function() rexp(m, 1), mon=mon) %>%
    run(until=n)
}
```

With the selected arrival rate, $\lambda = 1$, this test simulates an average of `n` arrivals entering a nearly saturated system ($\rho = 1/1.1$). Given that `simmer` generators are able to create arrivals in batches (i.e., more than one arrival for each function call) for improved performance, the parameter `m` controls the size of the batch. Finally, the `mon` flag enables or disables monitoring.

Let us build now the equivalent model using `SimPy`, with base Python for random number generation. We prepare the Python benchmark from R using the `rPython` package[3] as follows:

[3] C. J. G. Bellosta. *rPython: Package Allowing R to Call Python*, 2015. URL `https://CRAN.R-project.org/package=rPython`. R package version 0.0-6

```
rPython::python.exec("
import simpy, random, time

def test_mm1(n):
  def exp_source(env, lambd, server, mu):
      while True:
          dt = random.expovariate(lambd)
          yield env.timeout(dt)
          env.process(customer(env, server, mu))

  def customer(env, server, mu):
      with server.request() as req:
          yield req
          dt = random.expovariate(mu)
          yield env.timeout(dt)

  env = simpy.Environment()
  server = simpy.Resource(env, capacity=1)
  env.process(exp_source(env, 1, server, 1.1))
  env.run(until=n)

def benchmark(n, times):
  results = []
  for i in range(0, times):
    start = time.time()
    test_mm1(n)
    results.append(time.time() - start)
  return results
")
```

Equivalently, this can be done for SimJulia using the rjulia package[4]. Once more, n controls the average number of arrivals:

[4] Y. Gong, O. Keys, and M. Maechler. *rjulia: Integrating R and Julia – Calling Julia from R*, 2017. URL https://github.com/armgong/rjulia. R package version 0.9-3

```
rjulia::julia_init()
rjulia::julia_void_eval("
using SimJulia, Distributions

function test_mm1(n::Float64)
  function exp_source(env::Environment, lambd::Float64,
                      server::Resource, mu::Float64)
    while true
      dt = rand(Exponential(1/lambd))
      yield(Timeout(env, dt))
      Process(env, customer, server, mu)
    end
  end

  function customer(env::Environment, server::Resource, mu::Float64)
    yield(Request(server))
    dt = rand(Exponential(1/mu))
    yield(Timeout(env, dt))
    yield(Release(server))
  end

  env = Environment()
  server = Resource(env, 1)
  Process(env, exp_source, 1.0, server, 1.1)
  run(env, n)
end

function benchmark(n::Float64, times::Int)
  results = Float64[]
  test_mm1(n)
  for i = 1:times
    push!(results, @elapsed test_mm1(n))
  end
  return(results)
end
")
```

It can be noted that in both cases there is no monitoring involved, because either SimPy nor SimJulia provide automatic monitoring as simmer does. Furthermore, the resulting code for simmer is more concise and expressive than the equivalent ones for SimPy and SimJulia, which are very similar.

We obtain the reference benchmark with `n=1e4` and 20 replicas for both packages as follows:

```
n <- 1e4L
times <- 20

ref <- data.frame(
  SimPy = rPython::python.call("benchmark", n, times),
  SimJulia = rjulia::j2r(paste0("benchmark(", n, ".0, ", times, ")"))
)
```

As a matter of fact, we also tested a small DES skeleton in pure R provided in Matloff [2011, s. 7.8.3]. This code was formalised into an R package called `DES`, available on GitHub[5] since 2014. The original code implemented the event queue as an ordered vector which was updated by performing a binary search. Thus, the execution time of this version was two orders of magnitude slower than the other frameworks. The most recent version on GitHub (as of 2017) takes another clever approach though: it supposes that the event vector will be short and approximately ordered; therefore, the event vector is not sorted anymore, and the next event is found using a simple linear search. These assumptions hold for many cases, and particularly for this M/M/1 scenario. As a result, the performance of this model is only ~ 2.2 times slower than `SimPy`. Still, it is clear that pure R cannot compete with other languages in discrete-event simulation, and `DES` is not considered in our comparisons hereafter.

[5] https://github.com/matloff/des

Finally, we set a benchmark for `simmer` using the `microbenchmark` package[6], again with `n=1e4` and 20 replicas for each test. Figure C.1 (left) shows the output of this benchmark. `simmer` is tested both in monitored and in non-monitored mode. The results show that the performance of `simmer` is equivalent to `SimPy` and `SimJulia`. The non-monitored `simmer` shows a slightly better performance than these frameworks, while the monitored `simmer` shows a slightly worse performance.

[6] O. Mersmann. *microbenchmark: Accurate Timing Functions*, 2015. URL https://CRAN.R-project.org/package=microbenchmark. R package version 1.4-2.1

At this point, it is worth highlighting `simmer`'s ability to generate arrivals in batches (hence parameter m). To better understand the impact of batched arrival generation, the benchmark was repeated over a range of m values $(1, \ldots, 100)$. The results of the batched arrival generation runs are shown in Figure C.1 (right). This plot depicts the average execution time of the `simmer` model with (red) and without (blue) monitoring as a function of the generator batch size m. The black dashed line sets the average execution time of the `SimPy` model to serve as a reference.

The performance with `m=1` corresponds to what has been shown in Figure C.1 (left). But as m increases, `simmer` performance quickly

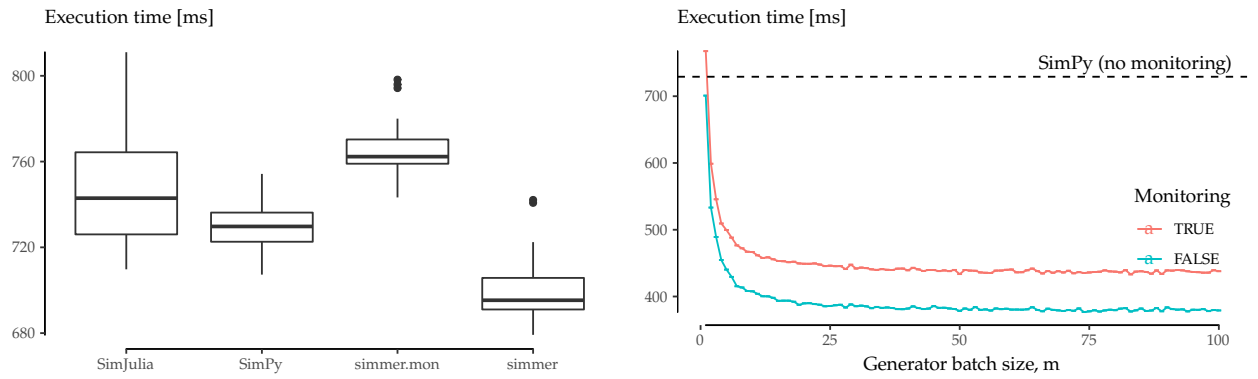


Figure C.1: Performance comparison. Boxplots for 20 runs of the M/M/1 test with `n=1e4` (left). Performance evolution with the batch size m (right).

improves and becomes $\sim$ 1.6 to 1.9 times faster than `SimPy`. Surprisingly, there is no additional gain with batches greater than 40-50 arrivals at a time, but there is no penalty either with bigger batches. Therefore, it is always recommended to generate arrivals in big batches whenever possible.

C.2 *The Cost of Calling R from C++*

The C++ simulation core provided by `simmer` is quite fast, as we have demonstrated, but performance is adversely affected by numerous calls to R. The practice of calling R from C++ is generally strongly discouraged due to the overhead involved. However, in the case of `simmer`, it not only makes sense, but is even fundamental in order to provide the user with enough flexibility to build all kinds of simulation models. Nevertheless, this cost must be known, and taken into account whenever a higher performance is needed.

To explore the cost of calling R from C++, let us define the following test:

```
library("simmer")

test_simmer <- function(n, delay) {
  test <- trajectory() %>%
    timeout(delay)

  env <- simmer() %>%
    add_generator("test", test, at(1:n)) %>%
    run(Inf)

  arrivals <- get_mon_arrivals(env)
}
```

This toy example performs a very simple simulation in which `n` arrivals are attached (in one shot, thanks to the convenience function `at()`) to a `test` trajectory at $t = 1, 2, ..., n$. The trajectory consists of a single activity: a timeout with some configurable `delay` that may be a fixed value or a function call. Finally, after the simulation, the monitored data is extracted from the simulation core to R. Effectively, this is equivalent to generating a data frame of `n` rows (see the example output in Table C.1).

Table C.1: Output from the `test_simmer()` function.

Name	Start time	End time	Activity time	Finished	Replication
testo	1	2	1	TRUE	1
test1	2	3	1	TRUE	1
test2	3	4	1	TRUE	1

As a matter of comparison, the following `test_R_for()` function produces the very same data using base R:

```
test_R_for <- function(n) {
  name <- character(n)
  start_time <- numeric(n)
  end_time <- numeric(n)
  activity_time <- logical(n)
  finished <- numeric(n)

  for (i in 1:n) {
    name[i] <- paste0("test", i-1)
    start_time[i] <- i
    end_time[i] <- i+1
    activity_time[i] <- 1
    finished[i] <- TRUE
  }

  arrivals <- data.frame(
    name=name,
    start_time=start_time,
    end_time=end_time,
    activity_time=activity_time,
    finished=finished,
    replication = 1
  )
}
```

Note that we are using a `for` loop to mimic the behaviour of `simmer`'s internals, of how monitoring is made, but we concede the

advantage of pre-allocated vectors to R. A second base R implementation, which builts upon the `lapply()` function, is implemented as the `test_R_lapply()` function:

```
test_R_lapply <- function(n) {
  as.data.frame(do.call(rbind, lapply(1:n, function(i) {
    list(
      name = paste0("test", i - 1),
      start_time = i,
      end_time = i + 1,
      activity_time = 1,
      finished = TRUE,
      replication = 1
    )
  })))
}
```

The `test_simmer()`, `test_R_for()` and `test_R_lapply()` functions all produce exactly the same data in a similar manner (cfr. Table C.1). Now, we want to compare how a delay consisting of a function call instead of a fixed value impacts the performance of `simmer`, and we use `test_R_for()` and `test_R_lapply()` as yardsticks.

The benchmark was executed with `n=1e5` and 20 replicas for each test. Table C.2 shows a summary of the resulting timings.

Table C.2: Execution time (milliseconds).

Expr	Min	Mean	Median	Max
`test_simmer(n, 1)`	429.8663	492.365	480.5408	599.3547
`test_simmer(n, function() 1)`	3067.9957	3176.963	3165.6859	3434.7979
`test_R_for(n)`	2053.0840	2176.164	2102.5848	2438.6836
`test_R_lapply(n)`	1525.6682	1754.028	1757.7566	2002.6634

As we can see, `simmer` is $\sim$ 4.4 times faster than `for`-based base R and $\sim$ 3.6 times faster than `lapply`-based base R on average when we set a fixed delay. On the ther hand, if we replace it for a function call, the execution becomes $\sim$ 6.5 times slower, or $\sim$ 1.5 times slower than `for`-based base R. It is indeed a quite good result if we take into account the fact that base R pre-allocates memory, and that `simmer` is doing a lot more internally. But still, these results highlight the overheads involved and encourage the use of fixed values instead of function calls whenever possible.

References

D. Agrawal. Analysis and optimization of energy efficiency in 802.11 distributed coordination function. In *IEEE International Conference on Performance, Computing, and Communications, 2004*, pages 707–712. IEEE, 2004. ISBN 0-7803-8396-6. DOI: 10.1109/PCCC.2004.1395159. p. 10

A. Akella, G. Judd, S. Seshan, and P. Steenkiste. Self-management in chaotic wireless deployments. In *Proceedings of the 11th Annual International Conference on Mobile Computing and Networking*, MobiCom '05, pages 185–199, New York, NY, USA, 2005. ACM. ISBN 1-59593-020-5. DOI: 10.1145/1080829.1080849. p. 82

P. Andres-Maldonado, P. Ameigeiras, J. Prados-Garzon, J. J. Ramos-Munoz, and J. M. Lopez-Soler. Optimized lte data transmission procedures for iot: Device side energy consumption analysis. In *2017 IEEE International Conference on Communications Workshops (ICC Workshops)*, pages 540–545, May 2017. DOI: 10.1109/ICCW.2017.7962714. pp. 96 and 97

K. Arras. An introduction to error propagation: Derivation, meaning and examples of cy = fx cx fx'. Sep 1998. Technical Report Nr. EPFL-ASL-TR-98-01 R3, Autonomous Systems Lab, EPFL, September 1998. p. 20

A. Azcorra, **I. Ucar**, A. Banchs, F. Gringoli, and P. Serrano. μNap: Practical micro-sleeps for 802.11 WLANs. *Computer Communications*, 110:175–186, Sept. 2017. ISSN 0140-3664. DOI: 10.1016/j.comcom.2017.06.008. pp. 63 and 102

A. Azcorra, **I. Ucar**, A. Banchs, F. Gringoli, and P. Serrano. Energy-saving method based on micro-shutdowns for a wireless device in a telecommunications network. WO/2018/015601, Jan. 2018. URL `http://www.google.com/patents/WO2018015601`. pp. 63 and 102

S. M. Bache and H. Wickham. *magrittr: A Forward-Pipe Operator for R*, 2014. URL `https://CRAN.R-project.org/package=magrittr`. R package version 1.5. p. 95

S. Baek, G. DeVeciana, and X. Su. Minimizing Energy Consumption in Large-Scale Sensor Networks Through Distributed Data Compression and Hierarchical Aggregation. *IEEE Journal on Selected Areas in Communications*, 22(6):1130–1140, Aug. 2004. ISSN 0733-8716. DOI: 10.1109/JSAC.2004.830934. p. 10

A. Bahouth, S. Crites, N. Matloff, and T. Williamson. Revisiting the issue of performance enhancement of discrete event simulation software. In *Simulation Symposium, 2007. ANSS '07. 40th Annual*, pages 114–122, March 2007. DOI: 10.1109/ANSS.2007.36. p. 13

V. Baiamonte and C.-F. Chiasserini. Saving Energy during Channel Contention in 802.11 WLANs. *Mobile Networks and Applications*, 11 (2):287–296, Mar. 2006. ISSN 1383-469X. DOI: 10.1007/s11036-006-4480-x. pp. 10 and 12

B. Balaji, B. R. Tamma, and B. S. Manoj. A Novel Power Saving Strategy for Greening IEEE 802.11 Based Wireless Networks. In *2010 IEEE Global Telecommunications Conference GLOBECOM 2010*, pages 1–5. IEEE, Dec. 2010. ISBN 978-1-4244-5636-9. DOI: 10.1109/GLOCOM.2010.5684071. pp. 12 and 52

N. Balasubramanian, A. Balasubramanian, and A. Venkataramani. Energy consumption in mobile phones: a measurement study and implications for network applications. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference - IMC '09*, page 280, New York, New York, USA, Nov. 2009. ACM Press. ISBN 9781605587714. DOI: 10.1145/1644893.1644927. p. 9

J. Banks. *Discrete-event System Simulation*. Prentice-Hall International Series in Industrial and Systems Engineering. Pearson Prentice Hall, 2005. ISBN 9780131446793. p. 90

L. A. Barroso and U. Hölzle. The Case for Energy-Proportional Computing. *Computer*, 40(12):33–37, Dec. 2007. ISSN 0018-9162. DOI: 10.1109/MC.2007.443. p. 35

N. Bartzoudis, O. Font-Bach, M. Miozzo, C. Donato, P. Harbanau, M. Requena, D. López, **I. Ucar**, A. Azcorra, P. Serrano, J. Mangues, and M. Payaró. Energy footprint reduction in 5G reconfigurable hotspots via function partitioning and bandwidth adaptation. In *2017 Fifth International Workshop on Cloud Technologies and Energy Efficiency in Mobile Communication Networks (CLEEN)*, pages 1–6, June 2017. DOI: 10.23919/CLEEN.2017.8045934. No citations

P. Basu and 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*,

IPSN '04, pages 196–204, New York, NY, USA, 2004. ACM. ISBN 1-58113-846-6. DOI: 10.1145/984622.984652. p. 4

D. Becam, P. Brigant, R. Cohen, and J. Szpirglas. Validité du modèle de neyman pour les processus d'erreurs sur des liaisons numériques à 2 et 140 mbit/s. In *Annales des télécommunications*, volume 40, pages 17–25. Springer, 1985. p. 55

C. J. G. Bellosta. *rPython: Package Allowing R to Call Python*, 2015. URL https://CRAN.R-project.org/package=rPython. R package version 0.0-6. p. 114

D. S. Berger, F. Gringoli, N. Facchi, I. Martinovic, and 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*, WiSec '14, pages 105–116, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2972-9. DOI: 10.1145/2627393.2627403. p. 57

S. Biaz and S. Wu. Rate adaptation algorithms for ieee 802.11 networks: A survey and comparison. In *2008 IEEE Symposium on Computers and Communications*, pages 130–136, July 2008. DOI: 10.1109/ISCC.2008.4625680. pp. 13 and 82

BIPM, IEC, IFCC, ILAC, IUPAC, IUPAP, ISO, and OIML. Evaluation of Measurement Data – Guide to the Expression of Uncertainty in Measurement, 1st edn. JCGM 100:2008. *Joint Committee for Guides in Metrology*, 2008. URL https://www.bipm.org/en/publications/guides/gum.html. p. 19

BIPM, IEC, IFCC, ILAC, IUPAC, IUPAP, ISO, and OIML. The International Vocabulary of Metrology – Basic and General Concepts and Associated Terms (VIM), 3rd edn. JCGM 200:2012. *Joint Committee for Guides in Metrology*, 2012. URL http://www.bipm.org/vim. p. 19

R. Bruno, M. Conti, and E. Gregori. Optimization of efficiency and energy consumption in p-persistent CSMA-based wireless LANs. *IEEE Transactions on Mobile Computing*, 1(1):10–31, Jan. 2002. ISSN 1536-1233. DOI: 10.1109/TMC.2002.1011056. p. 10

A. Carroll and G. Heiser. An Analysis of Power Consumption in a Smartphone. In *USENIX annual technical conference*, volume 14, pages 21–21. Boston, MA, 2010. pp. 10 and 30

M. Carvalho, C. Margi, K. Obraczka, and J. Garcia-Luna-Aceves. Modeling energy consumption in single-hop IEEE 802.11 ad hoc networks. In *Proceedings. 13th International Conference on Computer Communications and Networks (IEEE Cat. No.04EX969)*, pages

367–372. IEEE, 2004. ISBN 0-7803-8814-3. DOI: 10.1109/ICCCN.2004.1401671. p. 10

W. Chang. *R6: Classes with Reference Semantics*, 2017. URL https://CRAN.R-project.org/package=R6. R package version 2.2.2. p. 90

R. G. Cheng, C. H. Wei, S. L. Tsao, and F. C. Ren. Rach collision probability for machine-type communications. In *2012 IEEE 75th Vehicular Technology Conference (VTC Spring)*, pages 1–5, May 2012. DOI: 10.1109/VETECS.2012.6240129. p. 97

B. Cornaglia and M. Spini. Letter: New statistical model for burst error distribution. *European transactions on telecommunications*, 7(3): 267–272, 1996. p. 55

G. B. Creus and M. Kuulusa. Optimizing mobile software with built-in power profiling. In *Mobile Phone Programming*, pages 449–462. Springer, 2007. p. 9

M. Dong and L. Zhong. Self-constructive high-rate system energy modeling for battery-powered mobile systems. In *Proceedings of the 9th international conference on Mobile systems, applications, and services*, pages 335–348. ACM, 2011. p. 9

J.-P. Ebert, B. Burns, , A. Wolisz, and A. Wolisz. A trace-based approach for determining the energy consumption of a WLAN network interface. In *Proceedings of European Wireless*, pages 230–236, Feb 2002. p. 10

D. Eddelbuettel. *Seamless R and C++ Integration with Rcpp*. Springer-Verlag, New York, NY, USA, 2013. ISBN 978-1-4614-6867-7. p. 90

D. Eddelbuettel and R. François. Rcpp: Seamless r and c++ integration. *Journal of Statistical Software*, 40(8):1–18, 2011. DOI: 10.18637/jss.v040.i08. p. 90

M. Ergen and P. Varaiya. Decomposition of Energy Consumption in IEEE 802.11. In *2007 IEEE International Conference on Communications*, pages 403–408. IEEE, June 2007. ISBN 1-4244-0353-7. DOI: 10.1109/ICC.2007.73. p. 10

S. Eryigit, G. Gur, S. Bayhan, and T. Tugcu. Energy efficiency is a subtle concept: fundamental trade-offs for cognitive radio networks. *Communications Magazine, IEEE*, 52(7):30–36, July 2014. ISSN 0163-6804. DOI: 10.1109/MCOM.2014.6852080. p. 13

L. Feeney and 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*, volume 3, pages 1548–1557. IEEE, 2001. ISBN 0-7803-7016-3. DOI: 10.1109/INFCOM.2001.916651. pp. 2 and 10

D. Flater. Architecture for software-assisted quantity calculus. *Computer Standards & Interfaces*, 56:144–147, 2018. ISSN 0920-5489. DOI: 10.1016/j.csi.2017.10.002. p. 19

A. Garcia-Saavedra, P. Serrano, A. Banchs, and M. Hollick. Energy-efficient fair channel access for IEEE 802.11 WLANs. In *2011 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*, pages 1–9. IEEE, June 2011. ISBN 978-1-4577-0352-2. DOI: 10.1109/WoWMoM.2011.5986436. p. 10

A. Garcia-Saavedra, P. Serrano, A. Banchs, and M. Hollick. Balancing energy efficiency and throughput fairness in IEEE 802.11 WLANs. *Pervasive and Mobile Computing*, 8(5):631 – 645, 2012. ISSN 1574-1192. DOI: 10.1016/j.pmcj.2012.03.006. p. 13

K. Gomez, R. Riggio, T. Rasheed, D. Miorandi, and F. Granelli. Energino: A hardware and software solution for energy consumption monitoring. In *Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt), 2012 10th International Symposium on*, pages 311–317. IEEE, 2012. p. 10

Y. Gong, O. Keys, and M. Maechler. *rjulia: Integrating R and Julia – Calling Julia from R*, 2017. URL `https://github.com/armgong/rjulia`. R package version 0.9-3. p. 115

F. Gringoli and L. Nava. OpenFWWF - Open FirmWare for WiFi networks, 2015. p. 50

F. Gringoli, P. Serrano, **I. Ucar**, N. Facchi, and A. Azcorra. Experimental QoE evaluation of multicast video delivery over IEEE 802.11aa WLANs. *IEEE Transactions on Mobile Computing*, Early Access, 2018. ISSN 1536-1233. DOI: 10.1109/TMC.2018.2876000. No citations

P. Grover. Information Friction and Its Implications on Minimum Energy Required for Communication. *IEEE Transactions on Information Theory*, 61(2):895–907, Feb 2015. ISSN 0018-9448. DOI: 10.1109/TIT.2014.2365777. p. 1

L. Grupp, A. Caulfield, J. Coburn, S. Swanson, E. Yaakobi, P. Siegel, and J. Wolf. Characterizing flash memory: Anomalies, observations, and applications, 2009. ISSN 1072-4451. p. 31

A. Gupta and P. Mohapatra. Energy consumption and conservation in wifi based phones: A measurement-based study. In *2007 4th*

Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks, pages 122–131, June 2007. DOI: 10.1109/SAHCN.2007.4292824. p. 10

P. J. Havinga and G. J. Smit. Energy-efficient tdma medium access control protocol scheduling. In *Asian International Mobile Computing Conference, AMOC*, pages 1–10, 2000. p. 12

X. He and F. Y. Li. Throughput and energy efficiency comparison of one-hop, two-hop, virtual relay and cooperative retransmission schemes. In *2010 European Wireless Conference (EW)*, pages 580–587. IEEE, 2010. ISBN 978-1-4244-5999-5. DOI: 10.1109/EW.2010.5483469. p. 10

C. Hoymann, D. Astely, M. Stattin, G. Wikstrom, J. F. Cheng, A. Hoglund, M. Frenne, R. Blasco, J. Huschke, and F. Gunnarsson. Lte release 14 outlook. *IEEE Communications Magazine*, 54(6):44–49, June 2016. ISSN 0163-6804. DOI: 10.1109/MCOM.2016.7497765. p. 96

K. Huang, K. Duffy, and D. Malone. H-RCA: 802.11 Collision-Aware Rate Control. *Networking, IEEE/ACM Transactions on*, 21(4):1021–1034, Aug 2013. ISSN 1063-6692. DOI: 10.1109/TNET.2012.2216891. p. 13

T. Huehn and C. Sengul. Practical power and rate control for wifi. In *2012 21st International Conference on Computer Communications and Networks (ICCCN)*, pages 1–7, July 2012. DOI: 10.1109/ICCCN.2012.6289295. pp. 81 and 82

A. Hylick, R. Sohan, A. Rice, and B. Jones. An Analysis of Hard Drive Energy Consumption. In *2008 IEEE International Symposium on Modeling, Analysis and Simulation of Computers and Telecommunication Systems*, pages 1–10. IEEE, Sept. 2008. ISBN 978-1-4244-2817-5. DOI: 10.1109/MASCOT.2008.4770567. p. 30

IEEE. 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)*, pages 1–2793, March 2012a. DOI: 10.1109/IEEESTD.2012.6178212. p. 48

IEEE. IEEE Standard for Ethernet - Section 6. *IEEE Std 802.3-2012 (Revision to IEEE Std 802.3-2008)*, pages 1–0, Dec 2012b. DOI: 10.1109/IEEESTD.2012.6419740. p. 63

D. R. Irvin. Monitoring the performance of commercial t1-rate transmission service. *IBM journal of research and development*, 35(5.6): 805–814, 1991. p. 55

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, feb 2005. ITU-R Recommendation S.614. p. 55

ITU-R. P.1238: Propagation data and prediction methods for the planning of indoor radiocommunication systems and radio local area networks in the frequency range 300 MHz to 100 GHz. Recommendation P.1238, International Telecommunication Union, July, 2015. p. 72

K.-Y. Jang, S. Hao, A. Sheth, and R. Govindan. Snooze: energy management in 802.11n WLANs. In *Proceedings of the Seventh COnference on emerging Networking EXperiments and Technologies on - CoNEXT '11*, pages 1–12, New York, New York, USA, Dec. 2011. ACM Press. ISBN 9781450310413. DOI: 10.1145/2079296.2079308. p. 11

V. Jha and R. Bagrodia. Simultaneous events and lookahead in simulation protocols. *ACM Trans. Model. Comput. Simul.*, 10(3):241–267, July 2000. ISSN 1049-3301. DOI: 10.1145/361026.361032. p. 94

E.-S. Jung and 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*, volume 3, pages 1756–1764 vol.3, 2002. DOI: 10.1109/INFCOM.2002.1019429. p. 12

W. Jung, C. Kang, C. Yoon, D. Kim, and H. Cha. Devscope: a nonintrusive and online power analysis tool for smartphone hardware components. In *Proceedings of the eighth IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, pages 353–362. ACM, 2012. pp. 9 and 10

Jyh-Cheng Chen and Kai-wen Cheng. EDCA/CA: Enhancement of IEEE 802.11e EDCA by Contention Adaption for Energy Efficiency. *IEEE Transactions on Wireless Communications*, 7(8):2866–2870, Aug. 2008. ISSN 1536-1276. DOI: 10.1109/TWC.2008.070168. p. 10

G. Kalic, I. Bojic, and M. Kusek. Energy consumption in android phones when using wireless communication technologies. In *2012 Proceedings of the 35th International Convention MIPRO*, pages 754–759, May 2012. p. 9

A. Kamerman and L. Monteban. WaveLAN-II: A high-performance wireless LAN for the unlicensed band. *Bell Labs Technical Journal*, 2 (3):118–133, Summer 1997a. ISSN 1089-7089. DOI: 10.1002/bltj.2069. p. 12

A. Kamerman and L. Monteban. Wavelan®-ii: a high-performance wireless lan for the unlicensed band. *Bell Labs technical journal*, 2(3): 118–133, 1997b. p. 82

M. O. Khan, V. Dave, Y.-C. Chen, O. Jensen, L. Qiu, A. Bhartia, and S. Rallapalli. Model-driven energy-aware rate adaptation. In *Proceedings of the Fourteenth ACM International Symposium on Mobile Ad Hoc Networking and Computing*, MobiHoc '13, pages 217–226, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2193-8. DOI: 10.1145/2491288.2491300. pp. 5 and 13

D. Kim, W. Jung, and H. Cha. Runtime power estimation of mobile amoled displays. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2013*, pages 61–64. IEEE, 2013. p. 10

B. Lauwens. *SimJulia.jl: Combined Continuous-Time / Discrete-Event Process Oriented Simulation Framework Written in Julia*, 2017. URL `https://github.com/BenLauwens/SimJulia.jl`. Julia package version 0.3.14. pp. 13, 96, and 113

J. Lee, W. Kim, S.-J. Lee, D. Jo, J. Ryu, T. Kwon, and Y. Choi. An experimental study on the capture effect in 802.11a networks. In *Proceedings of the the second ACM international workshop on Wireless network testbeds, experimental evaluation and characterization - WinTECH '07*, page 19, New York, New York, USA, Sept. 2007. ACM Press. ISBN 9781595937384. DOI: 10.1145/1287767.1287772. p. 49

C.-Y. Li, C. Peng, S. Lu, and X. Wang. Energy-based Rate Adaptation for 802.11n. In *Proceedings of the 18th Annual International Conference on Mobile Computing and Networking*, Mobicom '12, pages 341–352, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1159-5. DOI: 10.1145/2348543.2348585. pp. 5 and 13

J. Liu and 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*, page 146, New York, New York, USA, June 2008. ACM Press. ISBN 9781605581392. DOI: 10.1145/1378600.1378617. p. 11

Luke Tierney. *A Byte-code Compiler for R*. Department of Statistics and Actuarial Science, University of Iowa, 2016. URL `https://www.stat.uiowa.edu/~luke/R/compiler/compiler.pdf`. p. 90

C. Mannweiler, M. Breitbach, H. Droste, I. L. Pavón, **I. Ucar**, P. Schneider, M. Doll, and J. R. Sanchez. 5G NORMA: System architecture for programmable multi-tenant 5G mobile networks. In *2017 European Conference on Networks and Communications (EuCNC)*, pages 1–6, June 2017. DOI: 10.1109/EuCNC.2017.7980662.

No citations

J. Manweiler and R. Roy Choudhury. Avoiding the rush hours: Wifi energy management via traffic isolation. In *Proceedings of the 9th international conference on Mobile systems, applications, and services*, pages 253–266. ACM, 2011. p. 10

N. Matloff. *The Art of R Programming: A Tour of Statistical Software Design*. No Starch Press, San Francisco, CA, USA, 1st edition, 2011. ISBN 1593273843, 9781593273842. p. 116

A. Mcgregor and D. Smithies. Rate adaptation for 802.11 wireless networks: Minstrel, 2017. URL `http://blog.cerowrt.org/papers/minstrel-sigcomm-final.pdf`. pp. 81 and 82

O. Mersmann. *microbenchmark: Accurate Timing Functions*, 2015. URL `https://CRAN.R-project.org/package=microbenchmark`. R package version 1.4-2.1. p. 116

National Instruments. Using a Unity Gain Buffer (Voltage Follower) with a DAQ Device. White paper, Jan. 2014. URL `http://www.ni.com/white-paper/4494/en/`. p. 18

J. Neyman. On a new class of 'contagious' distributions, applicable in entomology and bacteriology. *The Annals of Mathematical Statistics*, 10(1):35–57, 1939. p. 55

J. K. Nurminen and J. Noyranen. Energy-Consumption in Mobile Peer-to-Peer - Quantitative Results from File Sharing. In *2008 5th IEEE Consumer Communications and Networking Conference*, pages 729–733. IEEE, 2008. ISBN 1-4244-1457-1. DOI: 10.1109/ccnc08.2007.167. p. 9

A. J. Oliner, A. P. Iyer, E. Lagerspetz, S. Tarkoma, and I. Stoica. Collaborative energy debugging for mobile devices. In *HotDep*, 2012. p. 10

R. Palacios, F. Granelli, D. Gajic, C. Liß, and 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)*, pages 2443–2448, June 2013a. DOI: 10.1109/ICC.2013.6654898. p. 12

R. Palacios, F. Granelli, D. Kliazovich, L. Alonso, and 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)*, pages 1619–1625, Dec 2013b. DOI: 10.1109/GLOCOM.2013.6831305. p. 12

R. Palacios, G. M. Mekonnen, J. Alonso-Zarate, D. Kliazovich, and 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)*, pages 5941–5947, June 2015. DOI: 10.1109/ICC.2015.7249269. p. 12

R. Palacios-Trujillo, J. Alonso-Zarate, F. Granelli, F. H. P. Fitzek, and 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)*, pages 1–7, Dec 2015. DOI: 10.1109/GLOCOM.2015.7417689. p. 12

A. Pathak, Y. C. Hu, and M. Zhang. Bootstrapping energy debugging on smartphones. In *Proceedings of the 10th ACM Workshop on Hot Topics in Networks - HotNets '11*, pages 1–6, New York, New York, USA, Nov. 2011a. ACM Press. ISBN 9781450310598. DOI: 10.1145/2070562.2070567. p. 17

A. Pathak, Y. C. Hu, M. Zhang, P. Bahl, and Y.-M. Wang. Fine-grained power modeling for smartphones using system call tracing. In *Proceedings of the sixth conference on Computer systems*, pages 153–168. ACM, 2011b. p. 10

E. Pebesma and T. Mailund. *units: Measurement Units for R Vectors*, 2018. URL `https://CRAN.R-project.org/package=units`. R package version 0.5-1. p. 20

E. Pebesma, T. Mailund, and J. Hiebert. Measurement Units in R. *The R Journal*, 8(2):486–494, 2016. URL `https://journal.r-project.org/archive/2016/RJ-2016-061/index.html`. p. 20

X. Perez-Costa and D. Camps-Mur. IEEE 802.11E QoS and power saving features overview and analysis of combined performance [Accepted from Open Call. *IEEE Wireless Communications*, 17(4):88–96, Aug. 2010. ISSN 1536-1284. DOI: 10.1109/MWC.2010.5547926. p. 4

R. Prasad, A. Kumar, R. Bhatia, and 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), pages 1–6. IEEE, Jan. 2014. ISBN 978-1-4799-2317-5. DOI: 10.1109/CONECCT.2014.6740349. pp. 12 and 52

D. Qiao, S. Choi, and K. Shin. Goodput analysis and link adaptation for IEEE 802.11a wireless LANs. *Mobile Computing, IEEE Transactions on*, 1(4):278–292, Oct 2002. ISSN 1536-1233. DOI: 10.1109/TMC.2002.1175541. pp. 68 and 70

D. Qiao, S. Choi, A. Jain, and K. G. Shin. Miser: an optimal low-energy transmission strategy for ieee 802.11 a/h. In *Proceedings of the 9th annual international conference on Mobile computing and networking*, pages 161–175. ACM, 2003. p. 10

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2017. URL `https://www.R-project.org/`. pp. 17 and 89

A. Rice and S. Hay. Measuring mobile phone energy consumption for 802.11 wireless networking. *Pervasive and Mobile Computing*, 6(6):593–606, Dec. 2010. ISSN 15741192. DOI: 10.1016/j.pmcj.2010.07.005. p. 10

M. Richart, J. Visca, and J. Baliosian. Rate, power and carrier-sense threshold coordinated management for high-density ieee 802.11 networks. In *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 139–146, May 2015. DOI: 10.1109/INM.2015.7140286. pp. 82 and 83

R. Riggio, C. Sengul, K. Mabell Gomez, and T. Rasheed. Energino: Energy saving tips for your wireless network. In *Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication*, pages 273–274. ACM, 2012. p. 10

R. Rönngren and M. Liljenstam. On event ordering in parallel discrete event simulation. In *Proceedings of the Thirteenth Workshop on Parallel and Distributed Simulation*, PADS '99, pages 38–45, Washington, DC, USA, 1999. IEEE Computer Society. ISBN 0-7695-0155-9. p. 94

K. Samdanis, P. Rost, A. Maeder, M. Meo, and C. Verikoukis. *Green Communications: Principles, Concepts and Practice*. Wiley & Sons, 2015. ISBN 9781118759240. p. 1

N. Santhapuri, R. R. Choudhury, J. Manweiler, S. Nelakuduti, S. Sen, and K. Munagala. Message in message mim: A case for reordering transmissions in wireless networks. In *In HotNets*, 2008. p. 50

A. Schulman, D. Levin, and N. Spring. CRAWDAD data set umd/sigcomm2008 (v. 2009-03-02), Mar. 2009. URL http://crawdad.org/umd/sigcomm2008/. pp. 54 and 59

P. Serrano, A. de la Oliva, P. Patras, V. Mancuso, and A. Banchs. Greening wireless communications: Status and future directions. *Computer Communications*, 35(14):1651 – 1661, 2012. ISSN 0140-3664. DOI: 10.1016/j.comcom.2012.06.011. Special issue: Wireless Green Communications and Networking. p. 9

P. Serrano, A. Garcia-Saavedra, G. Bianchi, A. Banchs, and A. Azcorra. Per-Frame Energy Consumption in 802.11 Devices and Its Implication on Modeling and Design. *IEEE/ACM Transactions on Networking*, 23(4):1243–1256, Aug 2015. ISSN 1063-6692. DOI: 10.1109/TNET.2014.2322262. pp. 2, 3, 10, 23, 24, 25, 28, 29, 30, 33, 34, 36, 38, 40, 69, 71, 101, and 102

A. B. Sharma, L. Golubchik, R. Govindan, and M. J. Neely. Dynamic data compression in multi-hop wireless networks. *ACM SIGMETRICS Performance Evaluation Review*, 37(1):145–145–156–156, June 2009. ISSN 0163-5999. DOI: 10.1145/2492101.1555367. p. 10

E. Shih, P. Bahl, and M. J. Sinclair. Wake on wireless: an event driven energy saving strategy for battery operated devices. In *Proceedings of the 8th annual international conference on Mobile computing and networking - MobiCom '02*, page 160, New York, New York, USA, Sept. 2002. ACM Press. ISBN 158113486X. DOI: 10.1145/570645.570666. p. 10

S. Tarkoma, M. Siekkinen, E. Lagerspetz, and Y. Xiao. *Smartphone Energy Consumption: Modeling and Optimization*. Smartphone Energy Consumption: Modeling and Optimization. Cambridge University Press, 2014. ISBN 9781107042339. p. 9

Team SimPy. *SimPy: Discrete-Event Simulation for Python*, 2017. URL https://simpy.readthedocs.io/en/latest/. Python package version 3.0.9. pp. 13, 96, and 113

V. Tiwari, S. Malik, A. Wolfe, and M. Tien-Chien Lee. Instruction level power analysis and optimization of software. *Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology*, 13(2-3): 223–238, Aug. 1996. ISSN 0922-5773. DOI: 10.1007/BF01130407. p. 35

I. Ucar. *errors: Uncertainty Propagation for R Vectors*, 2018. URL https://CRAN.R-project.org/package=errors. R package version 0.2.1. p. 21

I. Ucar and A. Azcorra. Deseeding energy consumption of network stacks. In *IEEE 1st International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI)*, pages 7–16, Sept. 2015. ISBN 978-1-4673-8167-3. DOI: 10.1109/RTSI.2015.7325085. pp. 40 and 101

I. Ucar and B. Smeets. *simmer: Discrete-Event Simulation for R*, 2018. URL https://CRAN.R-project.org/package=simmer. R package version 3.7.0. p. 89

I. Ucar, A. Azcorra, and A. Banchs. Deseeding Energy Consumption of Network Stacks. In *6th Annual International IMDEA Networks Workshop*, June 2014. (invited poster). pp. 40 and 101

I. Ucar, C. Donato, P. Serrano, A. Garcia-Saavedra, A. Azcorra, and A. Banchs. Revisiting 802.11 Rate Adaptation from Energy Consumption's Perspective. In *Proceedings of the 19th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, MSWiM '16, pages 27–34, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4502-6. DOI: 10.1145/2988287.2989149. pp. 77 and 103

I. Ucar, C. Donato, P. Serrano, A. Garcia-Saavedra, A. Azcorra, and A. Banchs. On the energy efficiency of rate and transmission power control in 802.11. *Computer Communications*, 117:164–174, Feb. 2018a. ISSN 0140-3664. DOI: 10.1016/j.comcom.2017.07.002. pp. 87 and 103

I. Ucar, J. A. Hernández, P. Serrano, and A. Azcorra. Design and Analysis of 5G Scenarios with simmer: An R Package for Fast DES Prototyping. *IEEE Communications Magazine*, 56(11):145–151, Nov. 2018b. ISSN 0163-6804. DOI: 10.1109/MCOM.2018.1700960. pp. 99 and 104

I. Ucar, E. Pebesma, and A. Azcorra. Measurement Errors in R. *The R Journal*, 10(2):549–557, 2018c. ISSN 2073-4859. DOI: 10.32614/RJ-2018-075. pp. 28 and 101

I. Ucar, B. Smeets, and A. Azcorra. simmer: Discrete-Event Simulation for R. *Journal of Statistical Software*, (accepted for publication), 2018d. ISSN 1548-7660. URL https://arxiv.org/abs/1705.09746. pp. 99 and 103

N. Vaidya. An energy efficient MAC protocol for wireless LANs. In *Proceedings.Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 3, pages 1756–1764. IEEE, 2002. ISBN 0-7803-7476-2. DOI: 10.1109/INFCOM.2002.1019429. p. 10

L. Wang and J. Manner. Energy Consumption Analysis of WLAN, 2G and 3G interfaces. In *2010 IEEE/ACM Int'l Conference on Green Computing and Communications & Int'l Conference on Cyber, Physical and Social Computing*, pages 300–307. IEEE, Dec. 2010. ISBN 978-1-4244-9779-9. DOI: 10.1109/GreenCom-CPSCom.2010.81. p. 10

W. Wang, W. K. Leong, and 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*, WiNTECH '14,

pages 41–48, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-3072-5. DOI: 10.1145/2643230.2643231. p. 50

E. Weingartner, H. vom Lehn, and K. Wehrle. A performance comparison of recent network simulators. In *2009 IEEE International Conference on Communications*, pages 1–5, June 2009. DOI: 10.1109/ICC.2009.5198657. p. 13

H. Wickham and W. Chang. *ggplot2: Create Elegant Data Visualisations Using the Grammar of Graphics*, 2016. URL `https://CRAN.R-project.org/package=ggplot2`. R package version 2.2.1. p. 98

H. Wickham and R. Francois. *dplyr: A Grammar of Data Manipulation*, 2017. URL `https://CRAN.R-project.org/package=dplyr`. R package version 0.7.4. p. 98

S. H. Y. Wong, H. Yang, S. Lu, and V. Bharghavan. Robust rate adaptation for 802.11 wireless networks. In *Proceedings of the 12th Annual International Conference on Mobile Computing and Networking*, MobiCom '06, pages 146–157, New York, NY, USA, 2006. ACM. ISBN 1-59593-286-0. DOI: 10.1145/1161089.1161107. p. 82

Y. Xiao, R. S. Kalyanaraman, and A. Yla-Jaaski. Energy Consumption of Mobile YouTube: Quantitative Measurement and Analysis. In *2008 The Second International Conference on Next Generation Mobile Applications, Services, and Technologies*, pages 61–69, Sept 2008. DOI: 10.1109/NGMAST.2008.26. p. 9

F. Xu, Y. Liu, Q. Li, and Y. Zhang. V-edge: Fast self-constructive power modeling of smartphones based on battery voltage dynamics. In *nsdi*, volume 13, pages 43–56, 2013. pp. 9 and 10

L. Zhang, B. Tiwana, Z. Qian, Z. Wang, R. P. Dick, Z. M. Mao, and L. Yang. Accurate online power estimation and automatic battery behavior based power model generation for smartphones. In *Proceedings of the eighth IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, pages 105–114. ACM, 2010. p. 10

X. Zhang and K. G. Shin. E-MiLi: Energy-Minimizing Idle Listening in Wireless Networks. *IEEE Transactions on Mobile Computing*, 11(9):1441–1454, Sept. 2012. ISSN 1536-1233. DOI: 10.1109/TMC.2012.112. pp. 10, 11, 27, and 43

Y. Zhang, X. Wang, X. Liu, Y. Liu, Ł. Zhuang, and F. Zhao. Towards better cpu power management on multicore smartphones. In *Proceedings of the Workshop on Power-Aware Computing and Systems*, page 11. ACM, 2013. p. 10