Energy Efficiency in Large-scale Internet of Things Networks

Vilen Looga
Energy Efficiency in Large-scale Internet of Things Networks

Vilen Looga

A doctoral dissertation completed for the degree of Doctor of Science (Technology) to be defended, with the permission of the Aalto University School of Science, at a public examination held at the lecture hall T2 of the school on the 19th of June 2018 at 12 noon.

Aalto University
School of Science
Department of Computer Science
Distributed and Pervasive Computing
Supervising professor
Professor Antti Ylä-Jääski, Aalto University, Finland

Thesis advisors
Professor Zhonghong Ou, Beijing University of Posts and Telecommunications, China
Professor Yu Xiao, Aalto University, Finland

Preliminary examiners
Professor Paolo Bellavista, University of Bologna, Italy
Professor Hassan Charaf, Budapest University of Technology and Economics, Hungary

Opponent
Professor Jari Porras, Lappeenranta University of Technology, Finland

Aalto University publication series
DOCTORAL DISSERTATIONS 111/2018

© 2018 Vilen Looga

ISSN 1799-4934 (printed)
ISSN 1799-4942 (pdf)

Unigrafia Oy
Helsinki 2018

Finland
Abstract

Near-ubiquitous wireless connectivity combined with advancements in hardware and battery technology have enabled a proliferation of Internet-connected consumer and industrial devices. From smartphones in users' hands to Internet of Things (IoT) nodes streaming data from factory equipment, these devices have enabled new categories of services that have become a vital part of the consumer and industrial markets. Although various hardware and software aspects of wireless devices have seen immense improvements, battery technology still remains on a linear improvement path and optimization of the physical layer of the network stack is giving diminishing returns. Thus, the need to optimize the energy usage of the whole network stack still remains. This thesis focuses on the challenges related to understanding the effect of network traffic transmissions on the energy usage of the device. Specifically, research questions posited in this thesis look at the short-term predictability of user-driven network traffic and whether it can be exploited for traffic scheduling. Also, we explore what other variables affect energy usage during network transmissions and how they can be used in activity and energy models, which are needed to create less intrusive, leaner and more scalable energy profiling tools. For our research work, we chose two example scenarios - Android smartphones and IoT motes on a IEEE 802.15.4 network - each representing one of our focus networks.

Keywords Internet of Things, Energy Efficiency, Energy Modeling

ISSN (printed) 1799-4934 ISSN (pdf) 1799-4942
Location of publisher Helsinki Location of printing Helsinki Year 2018
Preface

I am forever grateful to my supervisor Prof. Antti Ylä-Jääski, who gave me the opportunity to work in one of the best research groups in Finland. His leadership of the group ensured that I and many other doctoral students could focus on doing our best possible work and never lack any resources necessary to get the job done.

I am thankful to my thesis instructor Prof. Zhonghong Ou, who patiently and firmly guided me to complete my work. His advice brought clarity in moments when my thoughts were anything but, and publishing the next paper seemed hopeless.

I would like to extend my special gratitude to Prof. Yu Xiao, who was my master's thesis instructor and whose work ethic and intelligence are a constant source of inspiration for me. Over the years, her advice has helped me to get over many bumps that I encountered on my road to the doctoral degree. Her feedback to the initial draft of this dissertation was invaluable.

I really appreciate the support of my frequent co-author Yang Deng, with whom I had many insightful discussions on the various aspects of IoT hardware and software. His nuanced understanding of the technology helped me build a solid technical foundation for my work.

A researcher thrives in an environment with excellent colleagues. Over the years I had the amazing opportunity to share office space and cups of coffee with many wonderful people, several of whom I can now consider my closest friends. Thank you Nie Pin, Aapo Kalliola, Miika Komu, Soili Feldt, Maarit Vuorio, Jaakko Kotimäki, Janne Savikko, Matti Siekkinen, Mario Di Francesco and Tuomas Aura. I would like to acknowledge my fellow doctoral students, with whom we shared the joys and difficulties of our chosen path: Jiang Dong, Marius Noreikis, Mark van Heejswik and Mohit Sethi. Those mentioned and many more, thank you for making the university such a wonderful place!

Many thanks to my pre-examiners Prof. Paolo Bellavista and Prof. Hassan Charaf who took the time and effort to review my thesis. Their suggestions helped us to ensure that the thesis meets the quality level expected in our community.

Prof. Jari Porras graciously accepted the role of the opponent for my defense
for which I thank him.

My warmest thanks to all of my friends here in Finland and all around the world.

As I have learned, doing a doctoral thesis is by no means an easy task. Quite often, it requires not only a sacrifice from you personally, but also the patience and support of the people who are the closest to you. I have the incredible luck to be surrounded by a loving family that has always encouraged me to pursue my own path in life. My dearest Mom and Dad, as with all my achievements, I dedicate this to you.

My lovely wife Sofia, you were the one that stood by me every day. Your encouragement and support helped me tremendously with the challenging task of writing this thesis. Your strength, intelligence and incredible optimism inspire me to be a better person every day. With you by my side, nothing is impossible. 
Obrigado!

Espoo, May 12, 2018,

Vilen Looga
Contents

Preface 1

Contents 3

List of Publications 5

Author's Contribution 7

1. Introduction 9
   1.1 Research Questions and Scope ............................ 10
   1.2 Methodology ............................................. 13
   1.3 Contributions ............................................ 15
   1.4 Structure ................................................ 16

2. Background and Related Work 17
   2.1 Energy Efficiency of Wireless Devices .................. 17
      2.1.1 Battery Life Still Matters ............................ 18
      2.1.2 Link Layer Protocols and Their Optimization ........ 19
      2.1.3 Network Traffic Prediction and Shaping ............... 25
      2.1.4 Energy Efficiency of Applications .................... 27
   2.2 Applications for Large-scale Internet of Things Networks . 30
      2.2.1 IoT Emulators and Simulators ......................... 30
      2.2.2 Large-scale IoT Testbeds ............................... 32
      2.2.3 Energy Modeling and Profiling of IoT Devices ........ 33
   2.3 Research Activities in IoT ............................... 34

3. Towards Energy Efficiency in Internet of Things Networks 35
   3.1 Energy Efficiency of Network Traffic Transmissions ...... 35
      3.1.1 Predictability of Network Traffic ..................... 37
      3.1.2 Traffic Scheduling .................................. 38
   3.2 Energy Modeling of Internet of Things Devices and Networks . 39
      3.2.1 IEEE 802.15.4 State Machine Energy Model ............ 41
      3.2.2 Mote Activity Classification .......................... 42
Contents

3.2.3   Emulation of Large-scale Internet of Things Networks  43
3.2.4   MAMMOTH Research Project  .............. 44
3.3     Open Questions and Future Work  .................. 44

4.      Conclusions  47

References  49

Errata  59

Publications  61
List of Publications

This thesis consists of an overview and of the following publications which are referred to in the text by their Roman numerals.


Author’s Contribution

Publication I: “The Great Expectations of Smartphone Traffic Scheduling”

The author of this thesis proposed to examine the assumption that smartphone network traffic is predictable and thus can be scheduled to increase the energy efficiency of wireless transmissions. This assumption is common in research related to smartphone traffic-scheduling schemes. The author analyzed a database of traffic traces from hundreds of smartphones collected by another research group. He identified which applications consume the most traffic and categorized them as system services or user-facing apps. The author evaluated the predictability of traffic traces from single apps and single-user devices using Augmented Dickey-Fuller test and ARIMA models.

Publication II: “Exploiting Traffic Scheduling Mechanisms to Reduce Transmission Cost on Mobile Devices”

The author of the thesis designed a constraint-solving algorithm to schedule network traffic from a smartphone into bursts, thus improving energy efficiency of transmissions. The author collected network traffic traces of various applications from an Internet tablet. Using an offline energy estimation model from the second author, the author evaluated energy savings achieved by the scheduling algorithm.

Publication III: “Inference-based Energy Modeling in Large-scale Internet of Things Testbeds”

The author of this thesis proposed a state machine-based energy model to evaluate the energy usage of Internet of Things devices based on their network
traffic. The author built a testbed with two different types of IEEE 802.15.4 motes and conducted experiments collecting wireless network traffic traces, while collecting power traces using the Monsoon Power Meter. He analyzed the energy usage estimated by his proposed energy model against the real world usage measured in motes.

Publication IV: “PowerShark: IEEE 802.15.4 Mote Activity Analysis Using Power Traces and Neural Networks”

The author of this thesis proposed a neural network-based classifier that could detect network activity and MAC layer type of a IEEE 802.15.4 mote based on the power usage. Using a testbed with several motes, the author collected power and traffic traces. He designed a neural network classifier and trained it with labelled power traces.

Publication V: “MAMMOTH: A Massive-scale Emulation Platform for Internet of Things”

The author of this thesis conducted an extensive survey of existing simulation and emulation tools for Internet of Things devices and networks. The survey found that most tools focus on small to medium scale evaluation, while a tool for large scale Internet of Things network is missing. This led the author, along with the other co-authors, to propose a design of a large-scale simulator based on virtual machines running in the cloud.
1. Introduction

We live in an unprecedented era, where instant access to global services and knowledge have become available to almost anyone anywhere. A fundamental part of that capability is near-ubiquitous wireless connectivity that has enabled Internet-connected consumer and industrial devices. From smartphones in consumers’ hands to wireless Internet of Things (IoT) nodes streaming sensor data from industrial machines, these completely untethered devices provide mobility and ease of installation thanks to the combination of battery power and wireless communications. Although immense improvements have been made to the various hardware and software aspects of wireless devices, the capacity of batteries has remained on a linear improvement path and optimization of physical layer of network stack is giving diminishing returns. Thus, the need for careful consideration of energy usage at all levels of the network stack still remains.

Energy efficiency problems are very acute when it comes to wireless devices [88]. In a modern smartphone, the wireless network interface (WNI) is the third-largest energy consumer after the display subsystem and microcontroller during active device use and the first during standby [19]. The same phenomenon is even more pronounced in case of constrained IoT devices, where the WNI can contribute over 90% of overall energy consumption [29]. However, compared to other hardware components, energy efficiency of the WNI might soon reach its limitation due to the physical properties of the radio antenna and the corresponding elements, including power amplifier, baseband processing circuits, among others. Thus, alternative approaches to improve energy efficiency are presently explored, among them are predicting and scheduling traffic [27].

To be able to understand the impact of network traffic on energy consumption of mobile devices, it is fundamental to gain insight into the characteristics of the network traffic shape, from both a single application and multiple applications perspective. Furthermore, as more and more sensors are integrated into mobile devices, it is essential to understand the energy consumption from the integrated sensors separately and their impact on the energy consumption of the host mobile devices. Taking a step further, as mentioned before, wireless devices naturally work in the form of networks, and therefore it is beneficial to view the energy
consumption of connected devices from the whole network perspective rather than single sensors [10]. This viewpoint not only enables deep understanding of the way devices interact with each other, and its impact on the corresponding energy consumption, but also possesses high potential to detect and pinpoint congestion occurring within the network [99]. Going further, accurate and low overhead methods to estimate and control energy usage of devices enable network-level energy management schemes, where a controlling entity can predict and dynamically adjust application behavior to balance between the requirements to data transmission parameters (velocity, frequency, volume etc.) and the battery life of devices in the network.

There are several challenges that need to be solved before energy management schemes can become useful. Firstly, several of the current methods of traffic prediction and scheduling introduce changes and significant overhead to the network stack (see Section 2.1.3). Secondly, the variety of wireless devices grows, as well as new or improved network layer protocols are introduced, especially on the MAC layer (see Section 2.1.2). An energy management scheme, for example, of IEEE 802.15.4 or IEEE 802.11 networks, needs to be able to estimate energy consumption of a device in a given application scenario and do that in a scalable and efficient way. Energy modeling is a common method to estimate energy consumption of protocols, components and devices. However, the challenge to model the behavior of various hardware and software combinations is non-trivial.

We identify modeling of relationship between network transmissions and energy usage as a critical issue, that needs to be addressed to enable efficient network level energy management. We posit the principal goals of this thesis, which are design and implementation of methods that enable accurate understanding of the energy usage on device and network level, without creating unnecessary overhead. In the following sections, we elaborate the scope and methodology of our work, and posit the core research questions that motivate it.

### 1.1 Research Questions and Scope

Improving energy efficiency on the network layer of IoT devices is a challenge that can be approached from many aspects. Whichever approach is chosen,
Introduction

Figure 1.1. IEEE 802.15.4 testbed: laptop with Ubuntu Linux for gathering network and power traces, and other experimental metadata; five Zolertia Z1s, one of them acting as a sink; four OpenMotes; Monsoon Power Monitor to measure the power consumption of the testbed or individual devices, as needed.

It is crucial that the experimental tools provide the necessary capabilities to practically test the hypothesis and the underlying assumptions. We focus on understanding the relationship between network transmissions and energy usage in devices participating in the prevalent IEEE 802.11 and IEEE 802.15.4 networks. Through theoretical discussion we have identified the following research questions in the problem domain:

**RQ 1.** Is the network traffic from the user-controlled devices predictable in the short term? At the time of our study there was no clear conclusion whether it is possible to predict traffic in multi-app usage scenarios, although several traffic-scheduling schemes were assuming some degree of predictability. Going further, can we exploit traffic predictability to improve energy efficiency of network transmissions? Current traffic-scheduling schemes have significant drawback, such as focusing on single-app scenarios and introducing significant overhead.

**RQ 2.** What are the variables and the relationships between them that affect the energy consumption of IEEE 802.15.4 devices during network transmissions? The foreseeable future is bringing us IoT networks with tens of thousands of devices and the IEEE 802.15.4 radio is the current prevalent standard. As energy usage is still a concern when developing IoT applications, multiple approaches for energy profiling have been developed for IEEE 802.15.4 devices, such as direct current measurements or estimation based on low-level system event logs. However, each of those approaches has some drawbacks. Thus, there is a
need for energy and activity models that would allow for less intrusive, smaller overhead and scalable energy-profiling tools.

Considering the scope of the research questions of this dissertation work, we chose two example scenarios (see Table 1.1), each representing one of our focus networks. These scenarios represent frequent real-world use cases and results from research could yield practical utilization methods. In the first scenario, we explore energy efficiency in modern smartphones, specifically Android phones, with multiple background software services and user-driven foreground applications. In the second scenario, we examine energy usage of a network of IoT motes equipped with IEEE 802.15.4 radio [6].

**Scenario I** is motivated by the ubiquitous usage of smartphones, which are becoming more powerful and feature-rich each year, while battery life still remains a concern. Significant improvements have been made to the hardware and software stack of smartphone devices, but an increasing amount of research work goes into examining the behavior of applications and how it affects the energy efficiency of the device. Network traffic from smartphone applications has a significant impact on energy usage. There exists an energy model that links network traffic to energy usage, providing a useful method for gaining insight into how application behavior impacts battery life [105]. Furthermore, traffic shaping has been proposed as a way to improve energy efficiency, which leads us to examine the possibility of traffic predictability and propose an algorithm to schedule the traffic for energy efficiency.

**Scenario II** is motivated by a similar idea to the one mentioned in Scenario I to understand the link between application behavior and energy usage, but applied to networks of Internet of Things devices. Over 20 billion of IoT devices are expected to be deployed by 2020 [36], mostly running in industrial settings. Most of these devices have low-power hardware running a single application, such as sensor reading or actuator control. Energy usage is very important in such scenarios, as IoT devices are expected to run for extended periods of time without needing maintenance work, such as replacing the battery. Additionally, it is expected that IoT networks will consist of heterogeneous devices running a large variety of use cases [77]. Developers and researcher alike need the ability to analyze how application behavior will affect energy usage not only on the device, but also on the network scale. Thus we propose to develop network traffic based energy model for IoT devices. Going further, we propose a simulator design that would enable to analyze how application behavior through network traffic impacts energy usage on the network scale.

As mentioned previously, due to large scope of the research questions, we needed to omit several topics that are also considered relevant. In our scenarios we did not include multi-functional end-user IoT devices, such as smart home hubs or voice-controlled appliances, which are not so common yet, although gaining popularity. In our research, we did not study improvements to the lower layers of the network stack, such as the design of a new MAC protocol for IEEE 802.15.4 radio or extensive tuning of an existing one. Instead, we chose
the existing popular MAC schemes and used them throughout our research. We did not investigate possible improvements to existing IoT simulators and emulators, rather we tried to thoroughly understand their possibilities and limitations. Finally, although methods developed in the course of our work enable the creation of network-level energy management schemes, we did not create any ourselves, as they are specific to IoT devices and applications. Thus, we leave that for future work.

1.2 Methodology

The research questions posited earlier are mainly concerned with finding causality or correlations among (a subset of) variables in the problem domain. Thus, our research methodology is motivated by the goal of accurate modeling of said relationships in the problem domain i.e modeling of the phenomenon [40, 26]. The main output of our research work, models created through empirical studies of the phenomenon, are practically applicable for further research as they address current concerns in the field. Guided by the principles of experimental computer science, our chosen methodology is summarized as follows, with descriptive questions and goals that each step of the study aims to achieve (see also Fig. 1.2):

- Theoretical discussion. We have chosen empirical studies as the foundation of our research work, thus we must decide which features of the phenomenon we want to model. The features of the model must be appropriate to answer the posited research question.

- Experimental framework. What quantitative data needs to be collected to do
Introduction

• Validation of the derived model. How is it different from reality, i.e. how accurately does it represent the phenomenon? What are its constraints and limitations?

• Analytical study. Descriptive discussion of the valid model (i.e answering the question "How?") might not be enough to answer the posited research question. Further analysis to discover the underlying causes of relationships exhibited by the model – i.e answering the question "Why?". What are the implications for future research work?

Having described our research methodology, we present more specific research methods used in our work below.

For experiments related to energy efficiency of network transmission on smartphones, we used a small-scale testbed consisting of a variety of Android phones, a router and a laptop. We collected traffic traces from the smartphone, while the Monsoon Power Monitor [7] was providing power through the battery pins of the phone. Before conducting experiments presented in this thesis, we measured the power consumption of various components of the phone to reduce the amount of noise in power traces.

For large-scale energy efficiency study we were able to get an early access to smartphone traffic traces collected by NetSense project at Notre Dame University [93]. NetSense project collected metadata of network traffic from hundreds of smartphones of university students during two intervals of several months. A data-logging app on the Android phones of the students collected minute-by-minute logs of how much each process transmitted and received data via WiFi or cellular networks. By analysing the data we were able to mathematically prove that certain applications do in fact exhibit predictability in the short term, whereas assumptions regarding overall smartphone traffic predictability were proven to be incorrect.

For IoT experiments investigating device behavior and power consumption we created a testbed consisting of several IEEE 802.15.4 motes (see Figure 1.1). A laptop connected to a sink node was used to manage the experiments and store data, while a Monsoon power monitor measured the power consumption of individual nodes or the whole testbed. On each of the motes we installed ContikiOS operating system with custom apps as needed. This testbed allowed us to conduct experiments to determine mote behavior and power usage in different scenarios, such as various emulated interference sources affecting wireless network transmissions. Using real hardware allowed us to get a more thorough understanding of how mote behavior with different MAC layer configurations and transmission types affects the power consumption. We verified the power estimates for radio and CPU provided by the manufacturers of the motes.
We designed packet-level algorithms to make wireless network transmissions more effective on smartphones and IoT devices alike. Through extensive study of experimental data we created and evaluated algorithms to efficiently model energy consumption of IoT devices. We designed and trained a neural network to efficiently and with no overhead provide non-invasive monitoring of IoT devices.

1.3 Contributions

This thesis is a summary of five peer-reviewed publications, each of them addressing the previously mentioned research questions. A brief overview of each publication is given in the following paragraphs.

Publication I looks at some of the underlying assumptions made frequently in smartphone traffic scheduling research. Specifically, we focus on assumptions that smartphone traffic can be predictable and scheduling traffic from a single app can have a positive effect on the energy efficiency of a single device. Using a large dataset of smartphone network traffic traces collected in the NetSense project, we conduct a statistical analysis of applications that have the most volume and the most transmissions. We look at short-term predictability of traffic from a single app and a single user (single device) using unit root and ARIMA models. We find that although background services, the largest network traffic sources, exhibit short-term predictability in some cases, most of the applications do not. Furthermore, we find that scheduling traffic from a single app does not improve the overall energy efficiency of the device, mostly because there is always unpredictable background traffic waking up the radio interface. Although smartphone software and hardware have developed significantly since the time the NetSense dataset was collected, the user behavior has remained much the same. Thus, we believe that our findings regarding user behavior impact on network transmissions efficiency are still relevant today.

Publication II proposes a network traffic scheduling algorithm to improve the energy efficiency of smartphone transmissions. We assume that network packets are transmitted as bursts with long intervals between bursts and shorter intervals between packets, and the existence of an oracle that knows short-term traffic schedule. To achieve energy efficiency, we propose to eliminate idle intervals within bursts and put the radio interface into sleep mode right after the end of each burst. The task of scheduling packets is transformed into constraint-scheduling problem that our algorithm can solve. We collect real traffic traces from smartphones and schedule the traffic with our algorithm. We use a previously developed packet-based WiFi energy model to estimate how much energy it would take a smartphone to send original and scheduled traffic. The results show significant savings when scheduling traffic with our algorithm. We leave prototype implementation of the scheduling on an Android phone for further work, as it would require designing additional software and significant kernel modifications.
Publication III presents a design, implementation and evaluation of a packet-based real-time energy model for Internet of Things devices. The generic model uses a MAC state machine to calculate the energy consumption of the device based on the sent and received packets. The model does not introduce additional overhead to the hardware or software of the devices, and can be easily extended with new devices and MAC layer protocols. We conduct extensive power consumption measurements using Monsoon power monitor with various devices and MAC protocol types to calculate the correct energy values for each state of the device. We evaluate the model by comparing the estimated power consumption to actual energy usage and find that our method offers high accuracy.

Publication IV proposes a way to infer information about IoT devices and their network activity by analyzing their power consumption. Based on a classical design of a neural network we implement a neural network classifier. We build a testbed with several IEEE 802.15.4 devices configured with ContikiOS and an application that exchanges random data with a sink. We collect and label power traces (measured with Monsoon power monitor) of IoT devices that are used as training data for the classifier. We train the classifier to recognize the type of the device, the MAC layer used and whether it is sending or receiving network packets. Evaluation shows that the classifier is able to recognize the correct device and its activity 95% of the time.

Publication V surveys the current state of Internet of Things emulators and simulators. We find that currently available tools are suitable for small- to mid-scale simulation of IoT networks and devices. Most of the available tools are simulators and focus on a specific aspect such as topology, machine code or radio interface. Distributed simulators are few and limited in their capabilities. Foreseeing large IoT networks with millions of participating nodes, we propose a design of a large-scale cloud-based distributed simulator that is capable of emulating millions of CoAP nodes and various radio protocols. The simulator is designed to simulate scenarios with millions of nodes connected over GPRS radio or proxies to a cloud-based middleware. The design presumes CoAP motes running in virtual machines with radio links emulated by ns2-like models. After assessing the necessary engineering work that would be needed to create a useful prototype of the emulator, we decided not to proceed with the implementation. Instead, we chose to use existing tools and testbeds for our experimental work (mentioned previously).

1.4 Structure

The thesis is divided as follows: Chapter 2 presents a survey of the state of the art in energy efficiency research and energy modeling in IoT and smartphones, mostly from the perspective of network transmissions. Chapter 3 gives a summary of the main contributions of this thesis. Chapter 4 concludes this thesis, followed by original papers attached to the end.
2. Background and Related Work

This chapter discusses in depth the background necessary to understand research scenarios and related works relevant to the research questions presented earlier.

2.1 Energy Efficiency of Wireless Devices

Extending battery life has been one of the key concerns of smartphone users since the emergence of these devices [78]. Lithium-ion batteries currently in use to power smartphones have constantly improved in regard to power and energy density. However, improvements have been linear and the technology itself is considered mature with an approaching need for significant improvements [25, 96]. On the other hand, features and computational power of smartphone devices have made significant leaps, which means that improvements in battery technology have not been used to significantly extend the usage time of devices, but rather to support design aesthetics (slimmer or more compact devices) and features. Thus, as study [44] found, battery life remains a big enough concern for smartphone users that they are willing to put a monetary value on the remaining charge.

From the perspective of energy efficiency another important aspect has appeared with the emergence of smartphone app stores. The focus of energy efficiency research has shifted from hardware and firmware optimization to studies of third-party applications and their effect on energy efficiency [64]. Since a significant part of installed applications use the wireless network interface (WNI) of the phone, which in itself is a large contributor to overall energy use of the smartphone [19], investigating their behavior has a great potential to improve the energy efficiency of the device.

We introduce the main challenges of energy efficiency that modern wireless devices, both smartphones and IoT motes, face in Section 2.1.1. We present energy-saving methods related to improvements of the physical and link layers of the network stack in Section 2.1.2 and discuss energy efficiency work related to traffic shaping and scheduling in Section 2.1.3. In concluding Section 2.1.4
we discuss work related to smartphone applications with the goal of improving energy efficiency. We omitted discussion of research work related to Physical Layer (PHY) of the network stack, such as radio wave modulation [87, 22] or network interface card optimization [100], since that is out of the scope of this thesis.

2.1.1 Battery Life Still Matters

One of the fundamental principles in human-centered product design of electronic devices, such as smartphones, is to abstract away the complexities of underlying hardware and software layers, and present a functional interface to the end user. The device should not bother the user with the intricacies of its internal working mechanisms, but offer useful interaction that near-seamlessly integrates into the day-to-day life of the user. Significant improvements have been made to various aspects of smartphone design, such as fast and widely available connectivity, and powerful hardware for useful and responsive user interfaces. However, batteries and battery life remain as one of the primary low-level aspects of technology that still surface and constrict the user experience of the device. Indeed, as several studies have shown [78, 44, 13, 47, 33], managing the battery life of the smartphone, for example, by decreasing the brightness of the screen or turning off wireless interfaces, is currently still something that most users have to do if they want to fully utilize their devices throughout the day.

In 2007, Rahmati et al. [78] first coined the term human-battery interaction (HBI) to describe cognitive and technical challenges of designing interfaces that would help users manage the inevitably limited battery life of their device. The study identified two types of users, one of which resorted to regular charging no matter the charge level of the battery, while the other used system indicators to decide whether it was time to charge the device. Based on the feedback from the participants the study identified several user interface issues (UI) that could be improved. First, the users perceived battery charge indicators as inaccurate, since they changed in a non-linear way. Secondly, power-saving settings were hard to find in the interface and provided inadequate information, for example by offering "Performance" or "Power-saving" toggle for WiFi without explaining clearly the implication of that choice. A more recent study [33] revisits findings of earlier studies and finds that most users are still struggling to understand how their device usage behavior affects battery, while some users still need to charge their device several times per day due to heavy use of applications. Follow-up study of users of crowdsourced smartphone battery-life awareness application Carat (also discussed in Section 2.1.4) [13] compiled based on their findings guidelines for energy awareness applications, such as giving as much information as possible about how particular applications drain the battery and providing clear actionable instructions on how to improve battery life by killing or replacing an app.
Concerns for battery life are alleviated by constantly increasing battery capacity, improving the energy efficiency of various software and hardware aspects, and influencing user behavior through better HBI design. Thus, there is a clear motivation for academic communities in fields such as wireless networking, mobile computing etc. to consider energy efficiency as one of the fundamental topics of research.

2.1.2 Link Layer Protocols and Their Optimization

Media Access Control layer (MAC), a part of the Link Layer of the wireless network stack, can be considered the lowest layer where it is feasible to optimize and to explore cross-layer solutions that take clues from application network traffic behavior. On the other hand, link layer protocols are developed for different wireless communication methods, such as cellular, WiFi or the IoT-centric IEEE 802.15.4, with each of them working in very different environmental conditions and performance envelopes.

**Long-Term Evolution (LTE)** is a cellular network standard managed by the 3GPP [101] that enables new services on handheld devices, such as high-definition media streaming, Voice-over-IP and high-mobility clients. The latest release of the standard called LTE Advanced (LTE-A) introduces several improvements specifically targeted for machine-to-machine (M2M) [53], vehicular [12] and device-to-device (D2D) [58] communication.

Upon powering on, a client (or user equipment (UE) as they are called in LTE specifications) tries to connect to a LTE network, by going through initial synchronization process, where it synchronizes to each frequency and checks whether it belongs to the right operator. Once at the right frequency, the UE receives the so-called Master and System Information Blocks to read various cell access parameters (we leave out the details of these steps for brevity).

Next comes the crucial step, when the UE actually tries to connect to the LTE network using Random Access (RA) procedure. There are two ways to do it based on the UE configuration, either contention-based or contention-free. In both cases, the UE is required to send an RACH preamble, which can have 64 possible signatures, over a shared medium. In contention-based approach reserved for delay-tolerant access requests, the UE chooses the preamble value randomly, which means that in case of multiple UEs with the same preamble
value connecting at the same time there might be collisions. Then it is up to the base station (called eNodeB in LTE specification) to send collision resolution message to one of the UEs to resume connection setup, while the other UE backs off for a later attempt to connect. In contention-free approach reserved for delay-constrained access requests, the network informs UE to use one of the RACH preamble values reserved specifically for this method [53]. Contention-free access is used for UE handover between base stations and fast resume of downlink transmissions.

Study [53] evaluated LTE RACH from the energy efficiency perspective in M2M scenarios, where over 1000 low-power devices are trying connect to a single base station. First, the study notes that current RA specification allows 10800 preambles per second (i.e. connection attempts) in an ideal scenario, which under realistic conditions is significantly lower due to contention-based access collision and random backoffs. Thus, even for realistic M2M scenarios with thousands of devices, the capacity of a single LTE base station might be limiting. The authors conduct simulated experiments configured with varying amount of RA slots per frame, simultaneous arrivals and maximum preamble re-transmission attempts per client. The results show that with a relatively small amount of simultaneous arrivals almost all configurations perform similarly offering energy-efficient access. On the other hand, with increased amount of arrivals per slot, energy consumption starts to increase significantly depending on the configuration. The authors conclude that energy-efficient LTE configuration for M2M scenarios is highly dependent on the application and consider several proposed improvements such as:

- optimizing MAC layer by embedding data into access process,
- using Access Class Mechanism (ACB) where devices are assigned a probability factor for access attempt based on their class,
- separation of RA resource by assigning RA slots to M2M or human-to-human (H2H) clients.
- as the most promising, Distributed Queuing [54], where logically distributed virtual queues are maintained by all devices in the network, with each device storing a partial representation containing the total amount of clients and its own position.

Predictive uplink resource allocation for M2M device groups [17] was proposed with the goal to reduce latency and thus save energy. Assuming that data transmissions are highly correlated within a group of event-driven devices, upon receiving a Scheduling Request (SR) from the first device in the chain, the base station can proactively allocate resources to other devices in the group ahead of their normal SR schedule. Simulated experiments conducted by the authors
Background and Related Work

with a 1D (chain) topology of 200 devices showed a 25% mean reduction in uplink latency.

Finally, both studies [58] and [12] point out that effective interference management, for example by allowing UE to control its transmission power when communicating directly to another device, is crucial in D2D and its subset V2V usage scenarios.

IEEE 802.11 (also known as WiFi) is a ubiquitous wireless connectivity standard [5] that since 1999 has used Power Saving Mechanism (PSM) to significantly reduce energy consumption of mobile devices connected to the network. WiFi interface has four basic modes of operation: idle, transmit, receive, sleep with significant differences in power consumption [105]. PSM is a cooperative mechanism that takes advantage of the low power mode of the WNI whereby the mobile client notifies the access point (AP) of its intention to go to sleep for a period of time (usually 100ms). The AP buffers frames destined to the client for the given period of time and broadcasts a beacon (Traffic Indication Map – TIM) to inform the client of the buffered frames. Once the client wakes up it receives the beacon and requests the buffered data from the AP. Compared to always on mode (CAM), PSM is approximately 20 times more energy-efficient [75]. One of the drawbacks of AP traffic buffering are the increased round trip times (RTT), especially in multimedia streaming scenarios with irregular traffic profiles [42], which may cause the sender to throttle TCP traffic and create noticeable latency for client apps. To overcome that, several adaptive PSM schemes have been proposed [75, 50, 95], with PSM-A being universally implemented in the firmware of WiFi chipsets and utilized by smartphone manufacturers. PSM-A extends the period of time the WNI stays active after a successful transmission, usually between 30-200 ms, which means any new frames from the stream can be received immediately. However, as modern streaming applications such as Netflix and Youtube start utilizing more bandwidth, the potential savings of PSM-A diminish, as the WNI is not able to go to sleep. Smart Adaptive Power Save Mode (SAPSM), a cross-layer approach proposed by [75], uses a machine-learning classifier that observes application traffic and assigns it a priority rating. Network traffic from applications classified as high priority is permitted to adaptively switch on CAM, while traffic from low-priority applications is always transmitted in PSM mode. SAPSM uses a classifier that has been trained on an assortment of popular applications with diverse network traffic profiles. Once an application has been classified, a kernel module intercepts outbound traffic and ensures that while network traffic from a certain application is transmitted, the WNI stays in CAM or PSM mode according to the priority rating of the app. For inbound traffic, SAPSM intercepts the incoming packet and compares the destination socket to the list of unique application UIDs. SAPSM was able to achieve up to 56% energy savings with two tested Android applications under normal usage patterns [75].

Another class of link-layer solutions for more efficient WiFi transmissions looks at reducing interference and channel access contention at the AP level
Authors of Network-Assisted Power Management (NAPman) were motivated by their findings that PSM implementation on most modern APs treat either CAM or PSM mode clients unfairly, and has negative impact on wireless capacity and energy usage. Depending on the implementation, the AP either schedules PSM packets at the end of the traffic queue (normal priority), which means CAM packets arriving later are transmitted first, or schedules PSM packet at the front of the queue (high priority), which means that CAM packets might have to wait for sleeping PSM clients to wake up and request their buffered packets first. In the former case, a single PSM client on the network has to stay in high-power CAM mode for longer, before all the other background traffic packets are transmitted, thus consuming more energy (2.5 times more compared to scenario with no background traffic, as one of the experiment by authors showed). On the other hand, in case of multiple PSM clients a high priority implementation causes clients to wait in high power CAM mode for the rest of the AP’s PSM packet queue to drain, spending up to 45% more energy per client. Thus, the authors of [81] propose NAPman energy-aware scheduling that ensures fairness. First, NAPman performs a fairness check on a PSM packet to decide whether it is fair to send it to the client on the next opportunity and only then advertises it to the client in the next beacon. Once the client notifies the AP that it is ready to receive the said packet, the AP moves the packets to the high-priority queue. Additionally, NAPman uses the ability of modern APs to create virtual APs and distributes PSM client evenly between them. Virtual APs transmit beacons at different times, which shortens the time interval a woken up PSM client has to wait in CAM mode for the (virtual) PSM queue to drain. Evaluation shows energy consumption of a single PSM client with NAPman is similar to high priority while ensuring fairness, whereas with multiple PSM clients NAPman clearly outperforms high-priority scheduling.

Finally, a new amendment to the standard was published in 2017 called IEEE 802.11ah (also named WiFi HaLow) [5], with improvements to range and energy efficiency specifically aimed at low-power IoT nodes [72]. For improved energy efficiency, IEEE 802.11ah introduces firstly beacon and page segmentation, which separates power-saving clients into different groups with shorter packet queues. Secondly, the new standard introduces a new power-saving mode in which buffering information is not included in the beacons for so-called non-TIM stations, i.e low-power clients that do not need to wake up frequently to receive beacons.

IEEE 802.15.4 [6] has emerged as the leading radio protocol developed with the specific goal to extend Internet communication architecture to the network stack of wireless sensors and objects – the Internet of Things, – while maintaining high reliability and low-power consumption [67]. IEEE 802.15.4 PHY layer operates on 2.4 GHz ISM band with a 250 kbps bitrate, 128 byte maximum packet size (including MAC, transport, routing and application headers) and a

1Industrial, scientific and medical bands that allow worldwide unlicensed use. There are some variations in channel availability depending on geographical location.
transmission range in hundreds of meters, while consuming 5-25 mA of current while transmitting or receiving packets (depending on the radio chip). Since 2010 IEEE 802.15.4 incorporated an energy-efficient MAC layer that uses Time Synchronized Channel Hopping (TSCH), where motes on the same network synchronize on a slotframe schedule (see Figure 2.1). The schedule indicates whether the mote should sleep or be active for transmitting or receiving in the given timeslot. Each active slot is long enough for sending a packet and receiving an ACK, and the schedule indicates which sender-receiver pairs can use it at their own channel offset. TSCH MAC itself does not specify how the schedule is built, but the two possible approaches are either a) centrally by a manager node that receives updated neighbor lists from participating nodes and sends back the new schedule, or b) by a distributed approach [8, 11], where nodes schedule links with their neighbors locally.

Although TSCH MAC protocol has become part of the IEEE 802.15.4 standard, there have been a few other energy efficiency-oriented MAC protocols that have found notable use, especially for experimental research work in academia. X-MAC [18] is an asynchronous protocol where the radio of each node has its own sleep schedule (duty cycle) with regular wake-ups for transmitting packets or checking for possible reception. If a sender wants to transmit a packet, it wakes up the radio interface from sleep mode and starts strobing a preamble until it gets interrupted by a woken receiver. ContikiMAC [28] introduces two main improvements to significantly enhance the energy efficiency of transmissions – phase lock and packet strobing. First, sender-receiver pairs detect each other's wake-up schedules and adjust them to minimize the strobing interval. Secondly, instead of sending a preamble, the sender starts strobing the packet immediately until receiving an acknowledgment (ACK) from the receiver or a timeout. These improvements make the ContikiMAC on average 2 times more energy-efficient in various scenarios as shown by our own experiments in Publication III.

There have been several notable technologies developed on top of the IEEE 802.15.4 standard. ZigBee [32] is a proprietary industry standard that defines enhancements for Network and Application layers, most important of them being routing and forwarding capabilities needed for mesh networking. Additionally, ZigBee comes with middleware and several applications designed for specific use cases, such as home automation, fitness monitoring and input device control. WirelessHART [92] is another industry-led protocol on top of IEEE 802.15.4 PHY layer developed specifically for industrial and automation control cases where compatibility with the HART communication protocol is needed. WirelessHART implements its own Time Division Multiple Access (TDMA) MAC layer with strict 10 ms timeslots, channel hopping and blacklisting for channels with consistent interference. 6LoWPAN [63, 52] is a protocol adoption that allows IEEE 802.15.4 to carry packets that use IPv6 addresses. 6LoWPAN utilizes header compression and address translation to fit the 128-bit IPv6 address in a packet and thus enable IEEE 802.15.4 devices to communicate over the newest IP standard. Finally, Constrained Application Protocol (CoAP) [89] is
an Application Layer protocol designed to bridge HTTP-based communication into IEEE 802.15.4 networks. CoAP enables transparent proxies that translate HTTP requests and responses to CoAP messages and back, thus allowing sensor motes to appear as endpoints with HTTP REST APIs. Initially CoAP was intended for communication over UDP packets, which do not have message ACK mechanism like TCP. Thus, CoAP specifies an ACK option in case the message sender wants to receive an acknowledgment from the receiver and a subscription mechanism that allows receiving multiple messages in a stream without needing to acknowledge every single one of them, for more energy-efficient communication.

**Bluetooth Low Energy (BLE)** [39] is a feature of the Bluetooth 4.0 Specification [91], a radio protocol for the 2.4 GHz ISM band that distinguishes itself from other IoT radio technologies with focus on single-hop short-range communication and application-specific profiles that participating devices must adhere to while maintaining battery life of several years. Compared to the classic Bluetooth protocol, defined in specifications 1.0-3.0, BLE uses a simplified communication stack (described below) that significantly reduces battery consumption, while maintaining the same transmission distance. The communication stack is divided into the Controller (usually implemented completely in hardware as a System-on-a-Chip) and the Host, where the former takes care of Physical and Link Layer functionality, while the latter handles logical links, service discovery, application profiles etc. BLE operates on 40 channels out of which 3 are used for advertising messages, while the rest are data channels. Devices with unidirectional network traffic e.g the ones that only transmit or receive (advertisers and scanners, respectfully) use the advertisement channels, while bi-directional connections utilize data channels. A bi-directional communication starts when a scanner responds to an advertisers message with a Connection Request (CR) message. An advertiser can accept multiple connections from scanners, while each scanner can have only one ongoing connection to an advertiser. Thus, the advertiser acts like a master in a tree topography piconet with one or more scanners as slaves. Each connection is identified by adding 32-bit access codes to packets. The master chooses a physical channel and uses a TDMA scheme to separate the channel into connection events. Each slave is assigned a channel hopping and wake-up schedule, thus allowing it to sleep and save energy between listening events. Packets include a More Data (MD) bit, that indicates whether there is more data waiting to be sent. Once both devices have no more data to send, the connection event closes and the slave device can sleep until the next event. According to [39], two parameters have the biggest impact on energy usage of a BLE device: connection interval and slave connection latency. The former specifies time interval between connection events in the schedule (7.5–4000 ms), while the latter tells how many connection events the slave can skip (0–499) before needing to wake up. In their evaluation, the theoretical battery life of a slave devices varied from a few days to over 12 years, based on connection interval of 7.5–4000ms and slave latency of 0, 1, 7 or 499 (maximum).
2.1.3 Network Traffic Prediction and Shaping

As we discussed previously, wireless devices benefit from putting their radio interface to sleep longer and organize their packet transmissions in a more energy-efficient way. We have discussed link layer solutions in Section 2.1.2 that generally try to optimize duty cycles of radio interfaces with the explicit goal of delivering the data if there are packets in the queue. Another class of solutions looks at higher-level, network packet scheduling problem that can be solved for optimal energy efficiency, for example by combining packets into tighter bursts and saving on tail energy [49, 56, 79]. Traffic-scheduling or traffic-shaping methods for wireless devices have been proposed with the goal to arrange uplink or downlink packets in the most energy-efficient way possible considering the contraints [35, 62]. In general, traffic-scheduling schemes either keep packets in the buffer to calculate energy-efficient transmission schedule or, in some cases, rely on traffic prediction methods and calculate the optimal future transmission schedule (more on that later). Scheduling schemes can be classified as offline in case all the traffic for the scheduling period is known, for example, when scheduling is done only on packets in the buffer. Online scheduling schemes work in real time as new packets arrive and calculate optimal schedule on the fly. For scheduling uplink packets, it is usually sufficient to run the scheme on the sender without controlling the receiver, since the sender knows which traffic needs to be sent. For scheduling downlink traffic, the scheduling scheme requires usually cooperation between the sender and the receiver. Finally, the primary target for scheduling schemes is traffic from media applications, such as video-streaming and video communication, due to their high overall traffic volume on mobile clients [42].

MoveRight scheme proposed by [35] buffers downlink traffic at the client and moves start of the packet transmission time until optimal point as calculated by energy cost function. The authors demonstrate that iterative local optimization of packet start times leads to globally optimal energy-efficient transmissions schedule.

CatNap [27] uses a proxy integrated into an Access Point (AP) that takes advantage of asymmetrical wired vs wireless transfer speeds to buffer traffic from data-oriented applications such as Web or File Transfer Protocol (FTP). Packets for delay-tolerant applications, arriving over relatively slower wired connection and destined for a wireless client, are buffered at the proxy until the client wakes up and then sent in a large chunk over much faster wireless connection, thus maximizing the time (up to 70% more) the radio interface of the client can stay in sleep mode. The success of CatNap is limited to how well it can recognize delay-tolerant traffic to ensure on-time arrival of packets and whether normal traffic is infrequent enough to let clients sleep. As our own work Publication I has shown, traffic from modern smartphones is almost always a mix from multiple application, with few of them delay-tolerant. Diadem [107] uses a proxy to pre-fetch data in batches for volume-heavy media streams.
Media applications can send requests for future data chunks that are buffered at the AP and transferred on the optimal time, thus ensuring on-time arrival of packets. The downside of pre-fetching approaches is that the buffered data might not be used, for example if the user stops watching the streamed video, thus creating significant traffic overhead. For this reason, service providers such as Youtube try to limit the amount of buffered data sent to the client. Hoque et al. [43] look at media stream TCP traffic shaping from the point of view of the service provider. Based on their previous work [41], the authors note that when streaming bursty TCP traffic the power consumption of the smartphone decreases as the received bursts get larger and as long as they can fit into the buffer of the client device. However, energy consumption of the client WNI increases if the burst is larger than the buffer. Thus, they implement EStreamer, a service provider’s side solution that finds the optimal burst size. It starts with Fast Start phase where content is initially buffered at maximum rate to overcome network jitter and bandwidth fluctuations, and then transitions to a binary search based optimal burst detection by increasing its size. The authors evaluate EStreamer by streaming videos from popular sites, such as Youtube and Dailymotion through the middleware. The results show 38-65% energy savings depending on the WNI used.

**Traffic prediction methods** try to predict packet transmissions in short or long-term time horizon, either on network or device-level. Traffic forecasting on the infrastructure level has been extensively explored by the research community, with the goal of improving resource reservation, anomaly detection and network monitoring, among others. Papadopoli et al. [69] looked at short-term traffic prediction in a IEEE 802.11 campus network. The authors implemented a system to poll wireless network AP every 5 minutes for traffic load information to collect historical data. Afterwards they tested various time series forecasting algorithms, such as *adaptive moving average*, *autoregressive integrated moving average (ARIMA)* and *historical mean hour-of-day*, depending on the forecast interval (5 minutes to week) and found that on finer time scales (5 minutes) the median traffic flow prediction error was 0.33 MB across all APs. Another work [23] looked at short-term online traffic forecasting for network links in cloud infrastructure, by applying a dynamic sliding window to various traffic predictors. The sliding window limited the amount of data that needed to be processed for traffic prediction and was able to significantly improve (7.6-101%) all tested predictors.

Wei et al. [103] observe that traffic flows between a client and a server during media streams contain no-data intervals that exhibit statistical size correlation to historical no-data intervals, thus being a potential way to proactively put the WNI to sleep for longer periods of time. The authors implement a proxy that monitors traffic flows between the media server and the client, and predicts the duration of the current sleep period using linear time series prediction approach. Unfortunately, the performance and achieved energy savings of the proxy are evaluated only by comparing linear prediction to historical mean average and
Background and Related Work

not baseline no-proxy scenario. Thus it is not possible to estimate energy savings achieved by sleep interval prediction compared to default setup. In a follow-up work [104] the authors implement the proxy on the client side and test it with various video formats. A power saving mechanism for WiMAX [59] uses Hidden Markov Model (HMM) to describe the current packet-level state of the network. HMM model has probabilities indicating the likelihood of moving from one state to another. The authors evaluate several prediction algorithms built on top of the HMM framework with resulting 5-25% in energy savings based on the traffic type.

2.1.4 Energy Efficiency of Applications

Research work on the energy efficiency of smartphone applications has recently focused on two new areas: using statistical analysis to find energy-sapping bugs in software and offloading computation from the mobile device to the cloud. The former gives a useful insight for app developers regarding potential bugs and informs users on apps draining their battery. The latter tries to automate the decision-making whether a computation should run on the device or it would be more energy-efficient to transfer the task to a cloud server.

Statistical analysis of application energy usage across various devices and firmware versions helps to uncover unexpected behavior, often software bugs in the application, manufacturer drivers or firmware itself, which causes an application to use significantly more energy. Carat [64] deployed a collaborative energy analysis tool on 409867 iOS and Android devices to look for anomalous energy drain caused by applications. Using the available battery usage APIs in the firmware, Carat collects energy usage of apps and compares it against a reference distribution. If running the app causes a significant energy drain compared to not running it, that app is classified as an energy hog. Such app may indeed require more computational resources on purpose to fulfill its task, thus it is up to the user to decide if they would continue using it. If an instance of the running app has a higher energy drain on a particular client configuration than on other configurations, that app is considered to have an energy bug. In total, over 100000 apps were analyzed, out of which 9.9% were found to be energy hogs with network connectivity being the main culprit of the increased energy drain. In addition, 233258 client-app instances were classified to have energy bugs, such as Kindle app draining significantly more energy when syncing over cellular network or Facebook Messenger being detrimental to battery life on some versions of Android. By following the recommendations of Carat, the users were able to save 11% of battery life over a two-week period, with a follow-up study [13] demonstrating a long-term effect on user behavior. An earlier large-scale study [65] of 20100 BlackBerry smartphone users over several months collected activity traces related to energy usage. Based on those traces the authors created an accurate model, which predicts whether an application can successfully finish execution before draining the battery. Additionally, the authors were able to
classify users based on their battery charging and discharging behavior. The classification method was incorporated into the Energy Management Oracle library, which can be polled in advance of application execution to determine whether the app finishes before draining the battery.

These statistical studies are complementary to a plethora of application energy analysis methods [109, 73, 60] that provide more in-depth tools for developers to look for potential energy bugs in their software. Eprof [73] is a tool for app developers that profiles energy use of an application at a fine-grained process, thread or routine level by monitoring system calls. In the first stage, Eprof is compiled into the app binary to trace (during runtime) system calls and app routines, which are then matched to each other. In the second stage, the routine is replayed, while system trace is used to drive a previously developed fine-grained Finite State Machine (FSM) energy model [74]. Thus, an application energy profile is created by matching routines to energy activities. A call graph of \((\text{routine, time, energy})\) tuples is created, which can demonstrate energy-sapping codepaths within the app. Authors evaluated Eprof on several popular applications and found several issues, such as an advertisement component within the Angry Birds game that consumed 45% of the app’s energy. A study done by the authors of Eprof also confirms that most of the energy is spent on I/O, specifically network transmissions. Most recently, eDoctor [60] aims to profile application phase changes with the goal of detecting abnormal battery drain, i.e. drain that is not caused by expected computationally heavy workload, but rather some underlying issue. eDoctor samples app activity, such as CPU utilization and use of sensors, at certain intervals and combines those intervals into distinguished execution phases, which are saved to per app resource usage tables. This information is combined with relevant events, such as update of the app or firmware, change of configuration etc. In case an abnormal battery drain is seen in the app, The Diagnosis Engine of eDoctor compares current execution phases with historic data to detect suspicious new phases with high energy usage. For example, eDoctor was able to detect a certain update to Facebook Messenger that contained a new execution phase where a wakelock was held while the app was idle, likely due to a bug in the code. The study found that among energy issues reported by users, over 90% were due to abnormal battery drain, with \(~30\%\) caused by bugs in applications.

**Offloading** software procedures for remote execution was proposed for more efficient resource allocation and communication in distributed environments [76, 45, 94, 82]. Recently, it has found traction in mobile cloud computing research due to its potential to speed up certain applications and save energy by offloading computationally intensive tasks from resource-constrained mobile phones to powerful cloud hardware. It should be noted that offloading differs from currently prevalent approach where mobile applications use REST APIs to do certain tasks, since the former is often done in an opportunistic way depending on the conditions, whereas the latter has an architecture where client and server-side code are clearly separated. Similar routine offloading has been
Background and Related Work

proposed for IoT scenarios [46] to dynamically switch application execution between the edge (also called Fog) and backend (Cloud) part of the network. However, so far no implementations of the idea could be found. CloneCloud [21] for Dalvik VM systems (Android) runs two clones of application-level virtual machines, one on the device and another in the cloud. The application itself does not require modifications; instead CloneCloud decides to dynamically move execution threads from the smartphone to the cloud clone and, after completion, re-integrate them back into the app execution on the device. The migrations points are decided during offline partitioning of the applications. First, a static analyzer finds legal partitions in the executable with certain restrictions, specifically allowing migration points to entry and exit points of Java methods, and narrowing the scope only to application methods. Secondly, partitions are profiled on different platforms using the dynamic profiler, which calculates their computational and energy costs. Finally, partitions (device or cloud clone) are chosen by the optimizer for the application binary and executed accordingly on runtime. Choosing which partitions and how often should be offloaded is crucial, as the energy cost of frequent communications can be detrimental to potential savings [51]. The authors evaluated CloneCloud by implementing three sample applications: a virus scanner, image search and targeted advertising. In each case, especially with larger workloads where the cost of migration was absorbed, offloading showed a significant speedup (10-20x) in computation and energy savings (9-20x) on the phone, both in case of 3G and WiFi connectivity. To further improve the performance of offloading, a new cloudlet-based approach [86] has been proposed. Cloudlets are VMs running on resource-rich devices, such as laptops or base stations, that are in close proximity to the mobile devices. Using nearby devices significantly increases bandwidth and decreases latency, while still offering significant speedup to computation [38]. A challenge with cloudlet-based offloading is intermittent connectivity, for example when the mobile client moves away while utilizing nearby cloudlets, thus losing offloaded tasks. To address this, an offloading optimization algorithm was proposed [110] that calculates the probability of success of offloading based on the mobility pattern of the user, local cloudlet availability and its admission policy. The optimization is formulated as a Markov Decision Process model, containing system states and actions. To obtain optimal offloading policy, the MDP problem is defined as a Bellman equation and solved by a value iteration algorithm. A more recent work [108] proposes an automated solution to offload Android applications to the cloud with method-level granularity, without need for prior (user generated) knowledge. The solution looks at the application structure and selects which methods (or entire classes) have dependencies, such as native or device specific libraries, that forbid them from being offloaded. Methods that pass all the checks are selected for offloading. The authors evaluated their solution with top 25 Google Marketplace apps and found that on average 24% of methods within an app could be offloaded.
2.2 Applications for Large-scale Internet of Things Networks

In this Section we will discuss some of the tools used for the design, development and evaluation of IoT technologies in industry and academia alike. We present emulators used for replicating the accurate behavior of hardware constrained devices, simulators for evaluating network traffic behavior in radiowave environment and testbeds that allow verification of software behavior on actual IoT hardware at scale. We summarize tools used in academia and industry to simulate IoT networks and emulate IoT devices in section 2.2.1, present the largest currently active IoT testbeds in section 2.2.2 and discuss some specific work related to the energy modeling of IoT devices in section 2.2.3.

2.2.1 IoT Emulators and Simulators

**COOJA** [66] is a cross-level simulator for the ContikiOS operating system, with the ability to simulate device behavior starting from machine code level, application behavior at system level and finally multi-device wireless communication at network level. Essentially, COOJA combines functionality previously provided by separate simulators such as Avrora [24], TOSSIM [55], NS-2 [14] and OMNeT++ [48] (see Fig. 2.2). COOJA includes a comprehensive GUI for creating and running simulations (see Fig. 2.3) and can provide network connectivity to the outside world through *tunslip* interface. Through a modular Java-based framework, COOJA has been extended by a number of hardware models, radio medium parameters and peripheral devices. During its lifetime, COOJA has been extended to run multiple operating systems and its network stack includes several new protocols such as IPv6, CoAP[30].

The simulated mote firmware is compiled and runs either directly on the CPU of the host machine or a MSP430 emulator, the latter being a MCU included in many hardware platforms popular in academia and industry alike, such as Zolertia Z1 and TelosB clones. In addition, it is possible to use simulated nodes written in Java for a faster simulation, which are suitable for higher-level algorithm testing, but do not produce deployable application code. COOJA is currently capable of running only on a single host machine (meaning no distributed
Background and Related Work

Figure 2.3. COOJA Graphical User Interface (GUI) running a simulation with dozens of nodes. Image courtesy of Wikimedia Commons, CC BY-SA 3.0 license.

simulations) and support up to a few hundred native nodes, due to simulation control code running in a single thread. It should be noted that although Contiki OS supports compilation for ARM-based platforms, COOJA still supports only emulation of MSP430 and STM32 architectures. Despite these and other limitations, such as inaccurate timing of TX transmissions in the MSP430 emulator [80], COOJA is still considered the most used simulation/emulation tool for IoT devices.

OpenWSN project [102] is the most comprehensive up-to-date open-source pure C implementation of various standardized IoT network stack protocols, including the new IEEE 802.15.4e standard. The project has ported the OpenWSN stack to various hardware platforms, ranging from MSP430 to ARM Cortex-M3, and provides the OpenSim tool for network simulation. OpenWSN architecture separates hardware platform specific code, such as drivers, into so-called board specific packages (BSP), while re-using the rest of the code in the network stack. The OpenSim replaces the BSP functions of OpenWSN with a special one that emulates node behavior on a regular PC. Each simulated node runs in its own process and communicates with Python-based controller thread via TCP session. Just as in COOJA, the simulation controller is event-based and its task is to synchronize step-by-step event execution of all participating nodes. The simulation can be paused at any moment to read its state. OpenSim also supports connectivity to real WSN nodes via TCP. The OpenWSN project also
includes OpenVisualizer GUI that shows real and emulated motes connected to
the host, their internal state (such as traffic counters) and the network topology,
while lacking features present in COOJA GUI, such as setting node location and
radio propagation models.

2.2.2 Large-scale IoT Testbeds

While emulators and simulators provide initial approximate estimation of per-
formance of the system at device and networks scales, testbeds are crucial for
real-world verification of proposed solution, especially with unexpected condi-
tions, such as node failures or interference [70]. Moreover, test facilities open
to all researchers provide a common baseline for repeatable and statistically
sound experimental results [71]. The inherent challenge to building a testbed
is that it should provide meaningful scale, thus enabling realistic and flexible
experimental scenarios. In some cases, such as vehicle-to-vehicle communication
scenarios [68], it might be quite difficult to achieve real world scale testing
environments. This has motivated the creation of many IoT-focused large-scale
testbeds [37], the largest two of them (as of writing of this thesis) we present
here.

FIT-IoT Lab [9, 70] is a large-scale open IoT testbed that is part of the OneLab
multi-site test facility [15] consisting of over 2700 wireless static and mobile
nodes connected to an Ethernet backbone for experiment control flow and teleme-
try transmission. The so-called open nodes have either MSP430 or Cortex-M3
microprocessors, use IEEE 802.15.4 radio and have several additional modules
and sensors such as GPS, temperature sensors, etc. Each open node is connected
to a gateway, which itself connects to the backbone, and a control node that is
used for reprogramming the open node and collecting telemetry, such as energy
consumption. The facility provides bare-metal access to the open nodes and
the gateways that connect them. Through web-based and command line tools
users can upload their own firmware binaries to the open nodes and monitor
various parameters during the experiments such as energy consumption, traffic
throughput and interference. On the software side, FIT-IoT Lab supports several
constrained operating systems such as ContikiOS, FreeRTOS and OpenWSN.

The SmartSantander project [85] developed a reference architecture for a
city-scale experimentation facility aimed at IoT research, with focus on such use
cases as environmental monitoring, outdoor parking management, participatory
sensing, etc. The reference architecture proposes a three-tier solution with
service, gateway and sensor nodes, each of them participating in the Testbed
Management and IoT Experimentation planes. Since the proposed infrastructure
is supposed to be installed in an actual city, it is designed so that smart city
services and experimental software can run at the same time. This way the
test facility can be used not only by the research community, but by other
organizations wishing to provision end-user smart city services. Currently, the
infrastructure of SmartSantader project is installed at four test sites in various

32
Background and Related Work

cities around Europe. At the Santander city test site alone, there are over 740 static nodes with a plethora of sensors for temperature, light, humidity and other measurements and over 150 mobile nodes installed on municipal vehicles. In addition, a participatory sensing smartphone app allows citizens of the city to submit localized and tagged images or text messages to enrich the data available for SmartSantander services.

2.2.3 Energy Modeling and Profiling of IoT Devices

In this section we briefly present some of the energy-profiling solutions aimed at IoT devices and networks.

PowerTrace [29] is an energy-profiling tool, available as part of the ContikiOS toolset, that tracks the power states of individual nodes and combines them into network-level accurate power estimation. The individual power state of the node is saved in so-called energy capsules, which store the energy spent by the node on a particular activity, such as transmitting packets or accessing data on flash memory, and duration according to hardware timers. Each energy capsule is associated with a particular application and protocol, thus allowing for accurate (94% accuracy according to authors) and granular energy estimation. PowerTrace is a software-based solution that has been implemented for the firmware of several physical motes and can be used within the COOJA emulator.

PowerTOSSIM [90] is an extension to TOSSIM, the TinyOS simulator, with the goal of providing node and hardware component-level power consumption estimation. First, the authors conducted microbenchmarks to get accurate power consumption of various hardware components in different statues, such as CPU per-instruction current and radio at different power levels. Based on the benchmark results, the authors develop hardware energy models. PowerTOSSIM takes advantage of TOSSIM's simulated hardware drivers, by augmenting them to send power state transition messages. These messages are used for offline analysis where time series of each hardware component’s power state transitions are given to the hardware energy models for accurate power estimation.

A more recent work focuses on energy profiling of devices in TSCH networks [98]. As mentioned earlier, TSCH MAC protocol uses synchronized timeslot schedule, where each node can perform a specific activity, such as sleeping, transmitting or listening. Based on the state of microcontroller and the radio (and the power draw of those states), the duration of the timeslot and the TSCH schedule, the energy model calculates energy consumption of the device during each of the timeslots. The authors evaluated their energy model with two types of physical motes and got encouraging results. Though the accuracy of estimated energy consumption for different types of timeslots was less than 95%, the energy estimation for entire slotframe was on average more than 90% accurate.
2.3 Research Activities in IoT

European Research Cluster on the Internet of Things (IERC) [3] is an umbrella organization for EU-funded IoT research projects with the goals of providing its participants a platform for communication, research coordination and development of overall vision for IoT-related research in EU. In total, over 65 research projects and several other consortia have participated in IERC during EU funding programs FP5-7 and Horizon2020. Each year IERC publishes an updated Strategic Research and Innovation Agenda (SRIA) [97], which gives an overview of the research activities of project participants and summarizes strategic directions of IoT research and development as seen by the stakeholders.

The 2017 edition of SRIA [97] mentions two specific research activities. Firstly, the **IoT European Platforms Initiative** [2], with projects focusing on open systems and platforms that can support applications for smart environments, businesses and services. The second initiative is the **IoT European Large-Scale Pilots Programme** [1], with projects focusing on deploying integrated IoT solutions at scale in various real-life scenarios. Each project has stakeholders from industrial partners across value-chain with tasks related to use, application, development, testing and integration activities.
3. Towards Energy Efficiency in Internet of Things Networks

In this chapter we demonstrate how through comprehensive design, implementation and evaluation work we address the core challenges posed by research questions in Section 1.1. For the context of Scenario I, we analyze traffic scheduling and develop our own algorithm, with the goal of improving energy efficiency. For Scenario II, we develop energy and activity models of IoT motes, aiming to enable leaner energy-profiling tools.

Starting with the smartphone scenario (Scenario I in Table 1.1), we address the challenges of RQ 1 with comprehensive traffic data analysis in Section 3.1.1 to study traffic predictability, followed by algorithm design for traffic scheduling in Section 3.1.2. We move to Scenario II to answer the questions posited in RQ 2 through the design and evaluation of IEEE 802.15.4 state-machine energy model in Section 3.2.1, followed by an activity analysis system in Section 3.2.2. We conclude the research part of the IoT section with a proposed architecture for a IoT testbed in Section 3.2.3 and a short description of the MAMMOTH research project in Section 3.2.4. The project started our research work in the field of IoT and the issues tackled in the project motivated most of our IoT-focused publications. We close this chapter with discussion on open research questions and future work in Section 3.3.

3.1 Energy Efficiency of Network Traffic Transmissions

In this section we explore the feasibility of improving energy efficiency of smartphone network transmissions using traffic prediction, as posited in RQ 1. Our study of the research question is performed in two parts. First, we evaluate the existing real-life smartphone traffic data for predictability in Section 3.1.1. We look into the type of processes and applications that send and receive network data and their transmission profile. Our findings show that some applications indeed exhibit short-term predictability, while total traffic from smartphones does not, due to its user-driven nature. This implies that the design of traffic-scheduling schemes relying on the assumption of traffic predictability must be carefully considered, since it is heavily dependent on the scenario. Following
Figure 3.1. ARIMA model burst size forecast error distribution for a single-user traffic trace. Sample size 50, mean burst size 465.4 kilobytes, forecast for next 5 bursts, forecast error mean 72.98 kilobytes.

Figure 3.2. Energy savings from traffic scheduling for different applications including randomly generated traffic.
that, in Section 3.2, we explore how to use traffic predictions to schedule network transmissions to be more energy-efficient, with the goal of scheduling traffic in a low-overhead minimally invasive way. Our proposed algorithm gathers network traffic into more consistent bursts and significantly improves the energy efficiency of wireless transmissions.

3.1.1 Predictability of Network Traffic

Our first task in Publication I was to survey the existing traffic-scheduling solutions for smartphone network traffic and find out what the common approaches and underlying assumptions are. From our extensive survey of the literature, we made two findings. First, most of the proposed solutions target a single application and do not schedule all of the traffic transmitted on the smartphone. Secondly, a significant amount of the proposed solutions make the assumption that traffic from a single app or the whole smartphone can be predictable in the short term. Since we could not find a study that would clearly argue for either of the assumptions, our second task was to study real smartphone traffic data and find out how these assumptions hold.

To that end, we got an early access to the smartphone traffic data collected by the NetSense [93] project, a two-year study with hundreds of participating Android smartphone users. The dataset consists of per-app per-user traffic counters that report transmitted data with 1-minute resolution. First, using energy models developed earlier, we estimated which apps have the biggest impact on the energy consumption of the device. We found that background services and the Facebook app (the dataset is from 2012) are the biggest contributors to energy consumption of the wireless interface, thus making them a good target for single-app traffic scheduling. Additionally, we looked at correlation between single-app traffic volume and burst frequencies in relation to impact on energy consumption. We found that apps with similar traffic volume could have orders-of-magnitude difference in energy impact, whereas the burst frequency of the app had a strong correlation with energy expenditure. These findings make a strong argument for improving energy efficiency of network transmissions by collecting traffic into fewer bursts.

The other part of our study is estimating whether traffic could be predictable either on single-app or the whole device level. We tested burst sizes and intervals of traffic traces with two methods frequently used for time series prediction: Augmented Dickey-Fuller (ADF) test [20] and ARIMA models [16]. The former determines whether a time series exhibits some long-term trends i.e is not essentially a random process, thus making it a potential candidate for short-term predictability testing with the latter. Out of all the tested single applications only 16% exhibited long-term trends, mostly background services. Out of those, none exhibited short-term predictability under various ARIMA models (see example Fig. 3.1). For multi-app (whole device) scenarios none of the traces exhibited long-term trends, most likely due to applications with unpredictable
traffic profiles dominating the overall traffic.

In conclusion, we identified good targets, such as background services and Facebook, among smartphone apps for improving energy efficiency of traffic transmissions. Our predictability evaluation using ADF and ARIMA shows that when it comes to user-driven single- and multi-app scenarios, the network traffic is not predictable in the short term. Our findings regarding application traffic predictability are by no means conclusive, but we believe that any future work should present a strong case for predictability before using it as an assumption.

### 3.1.2 Traffic Scheduling

Our work in Publication II is motivated by tail and intra-burst packet interval energy lost during wireless transmissions when the wireless interface stays in IDLE mode needlessly. The reason for such inefficiency is the burst-like profile of network traffic, where intra-burst packet intervals account for a large amount of time that the interface stays in IDLE mode. One of the solutions proposed to mitigate this issue is to collect traffic into more consistent bursts and allow the WNI to go into SLEEP mode more often, i.e. traffic scheduling. Our literature survey showed that at the time there were several traffic-scheduling schemes proposed for smartphones. Some of the traffic schemes relied on having previous knowledge of expected packet arrival times, either from statistical measurements or application hints. However, most of them focused on single-application traffic scheduling, while smartphones are used with multiple apps running at the same time in real life. Also, almost no solutions found in the literature were tested with random (i.e. unexpected) traffic.

Thus, we designed a traffic-scheduling algorithm that uses prior knowledge of packet arriving times and their priority based on the type of application that is sending it. Although some applications are more delay-tolerant than others, our algorithm would be limited to delaying the packet no more than 30ms. There are two reasons for this limitation. First, the user-facing applications will not exhibit any degradation in their usability with such a short delay. Secondly, as mentioned earlier, the biggest energy waste according to our estimate was happening in the intervals between packets belonging to the same burst. Thus, making the bursts more consistent should provide significant improvements to energy efficiency.

Our traffic scheduling algorithm was inspired by algorithms used for Constraint Solving Problems (CSP). Based on the packet priority, arrival time and the interval until the next packet, the algorithm would delay each packet, if necessary, and form more consistent bursts. We evaluated our algorithm with real traffic traces collected from a N810 smartphone for applications like Youtube and Internet Radio. For each traffic trace, the algorithm would generate a new traffic trace with scheduled traffic. We would compare the scheduled and unscheduled traces using an energy model. We found that depending on the application, the energy savings were up to 70%. For the second part of our evaluation, we added
random packets (Poisson process) to the scheduled trace to imitate background traffic that our algorithm was not able to schedule. Even with unexpected traffic, the energy savings were significant (see Fig. 3.2).

### 3.2 Energy Modeling of Internet of Things Devices and Networks

After exploring traffic scheduling for improving the energy efficiency of network transmissions in previous Section 3.1, we move to the challenges defined in RQ2 and the IoT network scenario. In Section 3.2.1 we present a lean energy-profiling solution for IEEE 802.15.4 motes, that can model the energy consumption of the device based on the network traffic trace. This solution demonstrates the strong connection between the network behavior of the applications running on the mote and the mote’s energy consumption. In Section 3.2.2, we present an activity classifier for the same type of motes. The classifier is able to accurately detect mote activity and configuration based on the power trace, thus demonstrating the significant and unique impact those variables have on mote’s power consumption. We conclude this part of our research work in Section 3.2.3 with a survey of existing emulation and simulation tools, and a proposal of an emulator design for a network of IEEE 802.15.4 motes running CoAP applications. Our design uses a distributed architecture to address scalability issues of current emulators and per VM emulated radio channels for flexible radio medium simulation.
Figure 3.4. An illustration of a power trace analysis using our classification pipeline. The trace is from an OpenMote with ContikiMAC transmitting packets. Each segment of the trace is colored according to the classification result. The example also demonstrates the difficulty of classification, as part of a broadcast transmission (blue) is classified incorrectly as Z1 broadcast.

Figure 3.5. List of emulators and simulators currently under active development and discontinued.

<table>
<thead>
<tr>
<th>Software</th>
<th>Last activity</th>
<th>Type</th>
<th>Network</th>
<th>Language</th>
<th>Supported protocols</th>
<th>Distributed</th>
</tr>
</thead>
<tbody>
<tr>
<td>NS</td>
<td>2012</td>
<td>SIM</td>
<td>Any</td>
<td>C, Python</td>
<td>All</td>
<td>NO</td>
</tr>
<tr>
<td>NS3</td>
<td>2012</td>
<td>SIM</td>
<td>Any</td>
<td>C, Python</td>
<td>All</td>
<td>NO</td>
</tr>
<tr>
<td>PDNS</td>
<td>2004</td>
<td>SIM</td>
<td>Any</td>
<td>C, Python</td>
<td>All</td>
<td>YES</td>
</tr>
<tr>
<td>GTNet5</td>
<td>2008</td>
<td>SIM</td>
<td>-</td>
<td></td>
<td>IEEE 802.11, IEEE 802.3</td>
<td>YES</td>
</tr>
<tr>
<td>J-Sim</td>
<td>2006</td>
<td>SIM</td>
<td>WSN</td>
<td>Java</td>
<td>-</td>
<td>NO</td>
</tr>
<tr>
<td>Jist</td>
<td>2004</td>
<td>SIM</td>
<td>Ad Hoc</td>
<td>Java</td>
<td>-</td>
<td>NO</td>
</tr>
<tr>
<td>TOSIM</td>
<td>2008</td>
<td>EMU/SIM</td>
<td>WSN</td>
<td>C [TinyOS]</td>
<td>-</td>
<td>NO</td>
</tr>
<tr>
<td>DSSimulator</td>
<td>-</td>
<td>SIM</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>NO</td>
</tr>
<tr>
<td>GlomoSim</td>
<td>2000</td>
<td>SIM</td>
<td>WSN</td>
<td>C, Parsec</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>OMNet++</td>
<td>2012</td>
<td>SIM</td>
<td>Any</td>
<td>C++, Java</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SensorSIM</td>
<td>2001</td>
<td>SIM</td>
<td>WSN</td>
<td>-</td>
<td>-</td>
<td>NO</td>
</tr>
<tr>
<td>SENSE</td>
<td>2008</td>
<td>SIM</td>
<td>WSN</td>
<td>C/C++</td>
<td>NullMAC, 802.11</td>
<td>NO</td>
</tr>
<tr>
<td>EMULAB</td>
<td>2012</td>
<td>EMU</td>
<td>Any</td>
<td>C/C++</td>
<td>Any</td>
<td>YES</td>
</tr>
<tr>
<td>ATEMU</td>
<td>2004</td>
<td>EMU/SIM</td>
<td>WSN</td>
<td>C</td>
<td>-</td>
<td>NO</td>
</tr>
</tbody>
</table>
3.2.1 IEEE 802.15.4 State Machine Energy Model

In Publication III we survey the existing energy-profiling tools for IEEE 802.15.4 and find that most solutions require either hardware access or modifications to the software. This motivates us to explore the feasibility of a packet-based energy model, such as the one developed in [105] for smartphone WNI. The benefit of a packet-based energy-profiling approach is that it can perform measurements by way of remote inference. This means that the profiling tool can be used in situations where it is not possible to install additional software or where hardware access is not feasible. The energy-profiling tool would only need the parameters of the device and its network stack and the traffic trace of the device, for example collected at the sink. Example scenarios where such energy-profiling tool could be used include lightweight testbeds and forensic work.

We propose an energy-profiling tool that uses a power state machine to model energy consumption of the mote based on the transmitted traffic. For each state we calculate the energy that is consumed by the device based on real-world measurements (more on that later). Once a packet is transmitted to or from the mote, the energy model adds energy spent during idle interval and packet energy to the power trace of the device. The energy model is defined in the following paragraph.

**Energy model**

Total energy $E_T$ spent by the mote is separated into radio $E_{RDIO}$ and the platform energy $E_{PT}$. Platform energy is separated into 3 categories:

- MCU in IDLE mode $E_{PTi}$ (time linear)
- MCU energy related to network activity $E_{PTn}$ (added to each packet)
- MCU other $E_{PTo}$ (modeled as a Poisson process $N(t)$)

Radio energy $E_{RDIO}$ consists of sent packets $E_{TX}$, received packets $E_{RX}$, ACK frames $E_{ACK}$, broadcast messages $E_{BC}$ and wake-ups $E_{WKP}$. Transmission and reception energy $E_{TX}$ and $E_{RX}$ are calculated by multiplying the time it takes to transmit the packet with power consumption of TX or RX mode, accordingly.

To estimate the energy used to transmit (TX) or receive (RX) a packet, we calculate the time it takes to keep the radio interface on (using bandwidth $b$) multiplied by the power consumption of TX or RX. We calculate the total energy consumption for a traffic trace with the length of $t$ seconds, where $n$ number of packets $p_i$ and $k$ number of broadcast messages were transmitted, and $m$ number of packets $p'_i$ received. For radio wake-up energy we divide the trace length $t$ with wakeup interval $t_{WI}$ and multiply by the length of wakeup beacon $t_{WB}$ and RX power.

Thus, we define the model as follows:
\[ E_T = E_{PT} + E_{RDIO} \]

\[ E_T = E_{PT_i} + E_{PT_r} + E_{PT_o} + E_{TX} + E_{RX} + E_{ACK} + E_{BC} + E_{WKP} \]

\[ E_T = P_{PT_i} \times t + N(t) + \sum_{i=1}^{n} \left( P_{TX} \left( \frac{\text{size}(p_i)}{b} \right) + E_{PT_i}(p_i) \right) \]
\[ + \sum_{i=1}^{m} \left( P_{RX} \left( \frac{\text{size}(p'_i)}{b} \right) + E_{PT_r}(p'_i) \right) + P_{TX} \times m \times \frac{\text{size}(p_{ACK})}{b} \]
\[ + P_{TX} \times t_{WI} \times k + P_{RX} \times \left( \frac{t \times t_{WB}}{t_{WI}} \right) \] (3.1)

**Evaluation**

For evaluation, we chose Zolertia Z1 and OpenMote hardware (testbed from Fig. 1.1) as the first devices to be modeled by our tool. We installed Contiki operating system on the devices and collected network traffic traces from experiments with ContikiMAC and X-MAC protocols. The experimental scenarios included sending UDP packets between leaf and sink nodes with 1-, 2- and 5-second intervals. From each experiment we collected the ground truth power trace using Monsoon Power Monitor and compared it to the estimate based on the traffic trace from our energy model. Based on the results (see Fig. 3.3), we can conclude that for every scenario our energy model is robust and accurate, with only slight underestimation of power consumption. The estimation error varied between 3-13%, with our model giving more accurate results for longer scenarios.

### 3.2.2 Mote Activity Classification

There is a need to understand the behavior of IEEE 802.15.4 motes, not only in the testing and deployment phase, but also later in their life cycle, i.e. when they are already installed and providing service. In certain production scenarios, such as smart homes or mining sites, it might not be easy or feasible at all to do forensic analysis of devices, for example due to limited access to hardware. This motivated us to develop a mote behavior analysis tool that requires only power trace from the power line to which the mote is connected to classify the hardware that is installed and its network transmissions. Similar approaches [84] have been used to identify home appliances and their activity based on the changes in the power draw on the main electric switch. We want to extend the same approach to include also IoT devices that usually have a very distinct power draw profile depending on the hardware platform, installed network stack.
and the state of the radio interface.

In Publication IV we present PowerShark, an activity classifier for IEEE 802.15.4 motes that uses power traces to detect the hardware type, MAC protocol and network transmissions of the participating motes. For the classification task we designed backpropagation neural networks, one using FANN and the other PyBrain libraries with slightly different learning algorithms. We chose neural networks for our classification task, because they are more robust when dealing with noise compared to Bayesian networks or Markov chains. To collect training data, we used our testbed with two OpenMotes and two Zolertia Z1s to collect ground truth of power and network traffic traces. By comparing the two traces we labeled samples in the power trace. Each label contained the type of hardware, installed MAC layer and the state of the radio interface. We used the labeled power traces to train our neural nets until they achieved over 95% accuracy both in training and testing. Our classifier is fast enough to work both offline and online. In the testing mode, the classifier takes short samples (125 ms sample length, default interval between wake-ups of the radio interface) of the power trace and assigns category with the highest confidence level.

Our work in classifying mote activity solely based on the power trace shows very encouraging results. Our classifier can reliably identify hardware types and radio state (see example classification results in Fig. 3.4). We believe even better results could be achieved, if the classifier worked on the whole power trace, not just single samples.

### 3.2.3 Emulation of Large-scale Internet of Things Networks

We are motivated by the increasing amount of IoT networks and their size, which require efficient tools to understand the the cross-layer issues that arise at such scale. Simulators and emulators for IoT devices are a cost-effective way to model the network infrastructure, evaluate performance of the network stack and analyze application behavior before going ahead with deployment.

In Publication V, our first task is to survey the existing IoT simulators (see Fig. 3.5 for summary). We found that most simulators focused on a specific aspect of the network. Indeed, the network simulators that provide the more general simulation capabilities combined with easy extendability, such as the ubiquitous NS-2, are the ones still under active development, whereas problem-specific simulation tools have seen little use beyond their originating work. Regarding scalability, some of the network simulators support distributed computing through MPI [34]. Among the surveyed emulators we find that the existing solutions provide a large variety of devices and aspects of modeling a system. However, all of them except EMULAB are able to run only on a single physical machine or virtual host (VM), mostly due to exponentially growing synchronization complexity for multiple emulated devices. This means that emulators are practically limited to a maximum few hundred devices. Thus, outside using testbeds with real devices, there is no way to emulate cross-layer application
behavior on a large scale. 

In the second part of our work in Publication V is a proposed design for an IoT testbed that would specifically address as accurate as possible energy simulation of IEEE 802.15.4 motes with CoAP applications participating in a heterogeneous network with GPRS, IEEE 802.15.4 networks and proxies bridging these networks with cloud services in the backend. Our design addresses the issue of scalability present in current emulators by distributing motes into multiple VMs, while allowing realistic and extendable way to model radio medium with emulated radio channels on each VM.

3.2.4 MAMMOTH Research Project

MAMMOTH was a 2-year research project starting from 2011 led by a consortium of Aalto and Oulu universities, and five industrial partners [4]. The main motivation for the project was to explore scalability of IoT architecture beyond millions of nodes. The project focused on industrial IoT networks with heterogeneous radio protocols and nodes that support RESTful protocols for communication and control. The main work packages (WP) of the project targeted scalability of protocols (WP1), security and congestion control (WP2), and user-facing services (WP3). The author of this thesis contributed to WP1 by conducting an extensive survey of existing simulation and emulation tools. Furthermore, the author proposed a novel emulator design. On the practical side, the author developed cloud-based CoAP node simulation tools for the project participants.

3.3 Open Questions and Future Work

In this section we will discuss some of the open questions left from our work and potential suggestions based on our results that could be useful for future work related to these topics.

IoT testbeds and similar experimental facilities for networking research are a crucial step in improving the reproducibility and reliability of experimental results. However, certain research work in networking, especially related to algorithm design (traffic scheduling schemes, MAC protocols, activity classifiers etc.), would greatly benefit from using publicly available datasets. For example, the machine learning community has adopted commonly used datasets, such as VOC [31] and ImageNet [83]. This way, the researchers do not have to collect their own data or wait to get access to a testbed, but can instead focus on improving the performance of their own solution and get immediate feedback how their work compares to the rest. Such datasets are larger than most research groups could assemble and provide a very clear way to compare performance among the competing solutions. For networking research, there are some available datasets, such as NetSense used in one of our publications. However the prevalent ap-
Towards Energy Efficiency in Internet of Things Networks

proach is still to use own datasets, either from simulation or collected from a testbed. If the dataset is not made public, it is difficult to assess for outsiders whether there is an unwanted bias in the data that could affect the outcome.

We acknowledge the profound effect that machine learning and specifically neural networks are currently having on the design of new solutions proposed in systems research relying on some form of signal analysis. The strength of machine-learning approach comes from the idea that any sufficiently complex neural net can be trained with enough data to achieve the same results as domain-specific signal analysis algorithms. Most prominently, this can be seen in computer vision research, where various neural nets are replacing classical computer vision algorithms. Secondly, a machine-learning approach greatly simplifies creating a solution to a signal processing task, as it requires only labeled data and a neural net designed by someone else (often already pre-trained on a large dataset). This means that for example a systems researcher does not need to delve into the intrinsic details of some classical signal-processing algorithm or do a comparative study to select the best one, but rather label data and train one of the available neural nets to solve the task. Thus, we believe that it is beneficial for most computer science researchers and especially young doctoral students to learn how machine learning can help them solve their tasks.

In our literature surveys we sifted through a lot of software tools developed for research work. Some of the tools have gained widespread usage, such as NS-2 and COOJA, while others are no longer developed. In our opinion, when research work requires implementation of some experimental tool, it is worth considering whether the tool should be developed as an independent software package or as an extension (plugin) to an existing framework. For example, we surveyed several network simulators that could have been implemented as a plugin to NS-2. Another example are device emulators that could be integrated into existing systems such as COOJA. This way the work of the researcher has a better chance for a long lifetime and a larger audience for a wider impact.
4. Conclusions

Our research work was motivated by the challenge of improving the energy efficiency of network transmissions in modern wireless devices. We wanted to gain a better understanding of how various phenomena affect the energy consumption of wireless transmissions on device and network level, specifically related to network traffic from user-driven applications in smartphones and service-driven applications in IoT networks. Our research was conducted according to methods of experimental computer science, mostly through empirical studies, with the goal of creating accurate models that would provide insight to our analysis.

To answer the posited research questions we chose two example scenarios – smartphones using WiFi and IoT motes in an IEEE 802.15.4 network – which represent the currently popular scenarios for energy efficiency research related to network transmissions. For smartphones, we explored the possibility to predict network traffic and schedule it in a more energy-efficient way with less overhead. For IEEE 802.15.4 motes, we created a lean energy model that describes the relationship between network transmissions and power usage. The second modeling tool that we created for IEEE 802.15.4 devices allows us to understand device activity based on the power draw alone. Finally, we proposed a new kind of scalable testbed architecture for large-scale IoT applications.

Our key findings demonstrate that for user-driven wireless network traffic it is possible to improve energy efficiency for certain applications, although the scope of traffic scheduling has to be chosen carefully. For service-driven network traffic, such as that present in IoT networks, we found a strong and quantifiable correlation between application behavior, mote configuration and energy consumption. These findings open up an opportunity to create new, behavior-oriented and leaner network-level energy management schemes.

We answered the posited research questions through thorough empirical studies and we hope that our results will give useful input to future research work. For example, on the practical side, creating scalable and extendable energy-profiling solutions for IEEE 802.15.4, IEEE 802.11 and other IoT networks. On a more general level, we hope that the results of our work demonstrate the power of creating useful abstract models through empirical studies and a new, behavior-focused way to look at energy research in wireless transmissions.
References


References


References


References


