Exploiting Energy Awareness in Mobile Communication

by

Ekhiotz Jon Vergara

Department of Computer and Information Science
Linköpings universitet
SE-581 83 Linköping, Sweden

Linköping 2013
Swedish postgraduate education leads to a Doctor’s degree and/or a Licentiate’s degree. 
A Doctor’s degree comprises 240 ECTS credits (4 year of full-time studies). 
A Licentiate’s degree comprises 120 ECTS credits.

Copyright © 2013 Ekhiotz Jon Vergara

ISSN 0280–7971 
Printed by LiU Tryck 2013

URL: http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-98656
Exploiting Energy Awareness in Mobile Communication

by
Ekhiotz Jon Vergara

October 2013
Linköping Studies in Science and Technology
Thesis No. 1627
ISSN 0280–7971
LiU–Tek–Lic–2013:61

ABSTRACT

Although evolving mobile technologies bring millions of users closer to the vision of information anywhere-anytime, device battery depletions hamper the quality of experience to a great extent. The massive explosion of mobile applications with the ensuing data exchange over the cellular infrastructure is not only a blessing to the mobile user, but also has a price in terms of rapid discharge of the device battery. Wireless communication is a large contributor to the energy consumption. Thus, the current call for energy economy in mobile devices poses the challenge of reducing the energy consumption of wireless data transmissions at the user end by developing energy-efficient communication.

This thesis addresses the energy efficiency of data transmission at the user end in the context of cellular networks. We argue that the design of energy-efficient solutions starts by energy awareness and propose EnergyBox, a parametrised tool that enables accurate and repeatable energy quantification at the user end using real data traffic traces as input. EnergyBox abstracts the underlying states for operation of the wireless interfaces and allows to estimate the energy consumption for different operator settings and device characteristics.

Next, we devise an energy-efficient algorithm that schedules the packet transmissions at the user end based on the knowledge of the network parameters that impact the handset energy consumption. The solution focuses on the characteristics of a given traffic class with the lowest quality of service requirements. The cost of running the solution itself is studied showing that the proposed cross-layer scheduler uses a small amount of energy to significantly extend the battery lifetime at the cost of some added latency.

Finally, the benefit of employing EnergyBox to systematically study the different design choices that developers face with respect to data transmissions of applications is shown in the context of location sharing services and instant messaging applications. The results show that quantifying energy consumption of communication patterns, protocols, and data formats can aid the design of tailor-made solutions with a significantly smaller energy footprint.

This work has been supported by the Swedish National Graduate School of Computer Science (CUGS), Sweden.

Department of Computer and Information Science
Linköpings universitet
SE-581 83 Linköping, Sweden
I would like to thank you,

for your

over this time.

Ekhiotz Vergara
Linköping, Sweden
October, 2013
# Contents

1 Introduction .................................................................................................................. 4  
    1.1 Preliminaries ........................................................................................................ 6  
    1.2 Problem formulation ............................................................................................ 7  
    1.3 Contribution .......................................................................................................... 8  
    1.4 List of publications .............................................................................................. 8  
    1.5 Thesis outline ....................................................................................................... 9  

2 Background ....................................................................................................................... 11  
    2.1 Application transmission energy .......................................................................... 11  
    2.2 Technological background ................................................................................. 13  
        2.2.1 3G energy consumption .............................................................................. 13  
        2.2.2 WiFi energy consumption ........................................................................ 17  
    2.3 Research directions ............................................................................................. 19  

3 EnergyBox .......................................................................................................................... 22  
    3.1 EnergyBox design .................................................................................................. 24  
        3.1.1 3G model .................................................................................................... 24  
        3.1.2 WiFi model ................................................................................................ 27  
    3.2 Evaluation ............................................................................................................. 28  
        3.2.1 Dataset and evaluation settings .................................................................... 28  
        3.2.2 Accuracy of EnergyBox 3G ........................................................................ 29  
        3.2.3 Accuracy of EnergyBox WiFi ..................................................................... 31  
    3.3 Limitations and discussion ................................................................................... 32  

4 Cross-Layer Burst-Buffering ............................................................................................. 33  
    4.1 Motivation ............................................................................................................. 34  
    4.2 Parameter inference algorithms ............................................................................ 34  
        4.2.1 Inactivity timer estimation algorithms ......................................................... 34  
        4.2.2 RLC data buffer threshold estimation .......................................................... 36  
        4.2.3 Evaluation of inference algorithms ................................................................. 40  
    4.3 Cross-layer burst buffering algorithm .................................................................. 41  
    4.4 Evaluation methodology ...................................................................................... 45  
        4.4.1 Algorithm implementation ............................................................................ 45  
        4.4.2 Energy assessment methodology ................................................................. 46
CONTENTS

4.4.3 Measurement-based methodology . . . . . . . . . . . . 48
4.5 Energy simulation results . . . . . . . . . . . . . . . . . . . 49
  4.5.1 Network transmissions energy savings . . . . . . . . 50
  4.5.2 CPU energy footprint . . . . . . . . . . . . . . . . . . 51
  4.5.3 Total energy savings . . . . . . . . . . . . . . . . . . 52
4.6 Measurement results . . . . . . . . . . . . . . . . . . . . . 53
4.7 Exploring live traffic . . . . . . . . . . . . . . . . . . . . . . 54
4.8 Discussion . . . . . . . . . . . . . . . . . . . . . . . . . . . 56

5 Energy-Efficient Location Sharing . . . . . . . . . . . . . . . 58
  5.1 MQTT protocol background . . . . . . . . . . . . . . . . . . 59
  5.2 Location sharing application . . . . . . . . . . . . . . . . . 61
  5.3 Evaluation methodology . . . . . . . . . . . . . . . . . . . . 64
  5.4 HTTP vs. MQTT . . . . . . . . . . . . . . . . . . . . . . . . . 65
  5.5 Protocol for check-in . . . . . . . . . . . . . . . . . . . . . 69
  5.6 Data encoding impact on HTTP . . . . . . . . . . . . . . . 70
  5.7 Summary and discussion . . . . . . . . . . . . . . . . . . . 71

6 Energy-Efficient Instant Messaging . . . . . . . . . . . . . . . 73
  6.1 Instant messaging dataset . . . . . . . . . . . . . . . . . . . 74
  6.2 Typing notification . . . . . . . . . . . . . . . . . . . . . . . 76
  6.3 Bundling of messages . . . . . . . . . . . . . . . . . . . . . . 77
  6.4 Evaluation environment and methodology . . . . . . . . . . 79
    6.4.1 Prototype IM application . . . . . . . . . . . . . . . . . 79
    6.4.2 Data set and parameter settings . . . . . . . . . . . . 80
  6.5 Energy cost of the typing notification . . . . . . . . . . . . 82
  6.6 Does message bundling pay off? . . . . . . . . . . . . . . . 84
    6.6.1 Energy savings . . . . . . . . . . . . . . . . . . . . . . . 84
    6.6.2 Message delay . . . . . . . . . . . . . . . . . . . . . . . 86
  6.7 Summary and discussion . . . . . . . . . . . . . . . . . . . . 88

7 Related Works . . . . . . . . . . . . . . . . . . . . . . . . . . . 89
  7.1 Measurement studies . . . . . . . . . . . . . . . . . . . . . . 89
  7.2 Energy saving techniques . . . . . . . . . . . . . . . . . . . 92
    7.2.1 Link selection techniques . . . . . . . . . . . . . . . . 92
    7.2.2 Low-level power saving . . . . . . . . . . . . . . . . . 95
    7.2.3 Traffic adaptation techniques . . . . . . . . . . . . . . 95
    7.2.4 Cooperation techniques . . . . . . . . . . . . . . . . . . 98
  7.3 Energy models . . . . . . . . . . . . . . . . . . . . . . . . . . 100

8 Conclusion and Future Work . . . . . . . . . . . . . . . . . . . . . 103
  8.1 Conclusions . . . . . . . . . . . . . . . . . . . . . . . . . . . . 103
  8.2 Future work . . . . . . . . . . . . . . . . . . . . . . . . . . . . 104

Appendices . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 107
A Energy Consumption of Recharging Smartphones 108
B Measurement setup 110
C Baseline comparison with IM applications 113
List of Figures

2.1 Transmission energy and standard deviation of different Instant Messaging applications for the short conversation. 12
2.2 Data pattern characteristics of the different Instant Messaging applications for the short conversation. 13
2.3 3G connection states. 14
2.4 Power consumption of different 3G states. 15
2.5 Power consumption of different 3G states for the operator 3. 17
2.6 Power levels of adaptive PSM for Samsung Galaxy SII and Sony Ericsson Xperia Arc. 18
2.7 Power levels of adaptive PSM for the Sony Ericsson Xperia Arc. 18
2.8 Taxonomy of wireless transmission energy at the user end. 19
3.1 Overview of EnergyBox function. 23
3.2 Overview of the EnergyBox state machines for 3G (a) and WiFi (b). 25
3.3 Measured time to empty the buffer $T_{e}$. 26
3.4 EnergyBox 3G accuracy for different traces. 30
3.5 A fragment of a 3G measurement and EnergyBox 3G inferred output for the Email trace. 30
3.6 A fragment of a 3G measurement and EnergyBox 3G inferred output for the Web1 trace. 31
3.7 EnergyBox WiFi accuracy for different traces. 31
4.1 The effect of small packet aggregation on the energy consumption. 35
4.2 RTT values obtained for $T_1$ inference. In the example InterPacket-Interval is set to 0.5 seconds and TestPeriod of $T_1$ is 7 seconds. 38
4.3 Example of lines 18-34 of Algorithm 2 showing the packets sent, their size and the state of the UE. 40
4.4 Comparison of algorithms in terms of energy consumption and elapsed time. 41
4.5 Characteristics of the emulated traces. 47
4.6 Measurement-based CPU model. 48
4.7 Overview of the TestFramework running in the test computer of the evaluation environment. 49
4.8 Network transmissions energy savings. 50
4.9 Percentage of time spent by the UE in the different 3G states over the total time of the original traces (y axis) scheduled with different $T_w$ (seconds) by the KLS (x axis). 51
4.10 Example of CPU energy consumption by running the KLS (left) and CPU energy footprint for the different traces (right). 51
4.11 Network transmissions energy savings, KLS energy overhead and total energy savings as a percentage of no-KLS base energy consumption. 52
4.12 Energy savings of cross-layer scheduling and TailEnder for operator 1. 53
4.13 Energy savings of cross-layer scheduling and TailEnder for operator 2. 55
4.14 Energy consumption examples of background traffic and energy-efficient shaped background traffic. 55

5.1 Overview of the data flow in a location sharing service based on cellular (3G) communication. 59
5.2 MQTT operation example. 60
5.3 Location Sharing Application architecture. 62
5.4 Location Sharing Application interaction. 62
5.5 Amount of data traffic and energy consumption for HTTP and MQTT in the idle state. 65
5.6 Amount of data traffic for HTTP and MQTT in active state. 66
5.7 Empirical CDF of packet size and inter-packet interval of HTTP and MQTT (9 users and T=15 s scenario). 67
5.8 Normalised energy consumption for HTTP and MQTT in active state. 68
5.9 Percentage of time spent by the UE in the different RRC states over the experiments for HTTP and MQTT. 69
5.10 Empirical CDF of inter-update interval in hours and minutes of the 22,387,930 check-ins from 224,804 users. 70
5.11 Normalised energy consumption and user equipment RRC states for different data encodings. 71

6.1 Empirical CDF of inter-message interval and message size of the dataset. 75
6.2 IM traffic of the dataset. 76
6.3 Architecture of the IM prototype implementation and the test environment. 79
6.4 Test conversations. 81
6.5 Normalised average energy and standard deviation of the typing notification feature for the different conversations. 83
6.6 Normalised average energy and standard deviation of the bundle technique for the different conversations. 84
6.7 Box plot of the delay experienced by the different messages for Sparse and Dense conversations. 87

7.1 Taxonomy of wireless transmission energy at the user end. 90
7.2 Traffic aggregation and batching. 98

A.1 Energy production of the world’s nuclear power plants. 109
B.1 Measurement setup for 3G and WiFi. 110
B.2 Aggregated power consumption for CPU and network transmission. 111
B.3 Stabilised CPU power trace and a network transmission. . . . . . . . 112

C.1 Energy consumption of the basic implementation compared to popular IM applications when sending the Short conversation. . . . . 113
C.2 Time spent in the different RRC states of the basic implementa-
tion compared to popular IM applications when sending the Short
conversation. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 114

List of Tables

2.1 Short conversation. . . . . . . . . . . . . . . . . . . . . . . . . . . . 12
2.2 State transitions and triggering packet sizes for TeliaSonera mea-
sured in July 2012. . . . . . . . . . . . . . . . . . . . . . . . . . . 16

6.1 Number of bundles and bundles per message for the Dense conver-
sation. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 85
6.2 Number of bundles and bundles per message for the Sparse conver-
sation. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 86
Chapter 1

Introduction

With the advent of computationally powerful handsets and mobile services we are finally on the verge of entering the era of “information anytime anywhere”. Wireless communication is becoming the dominant form of communication, and true mobility will increasingly be supported by widespread deployments of cellular infrastructures. Unfortunately, battery technology has not kept up with this evolution making the new power hungry capabilities also a hinder for future development. Mobile users still experience short battery lifetimes, making the energy consumption the Achilles’ heel of user quality of experience.

Ubiquitous connectivity and the current mobile data plans have led to a massive mobile data explosion and mobile traffic is forecasted to continue experiencing tremendous growth [1]. This growing wave of mobile data communication has several extreme consequences: (1) unforeseen data volumes make the mobile operators eager to squeeze every bit of the possible capacity and minimise the cost of their infrastructure, (2) the significant increase in the use of the radio hardware interfaces in mobile devices results in shorter battery lifetimes, and (3) the mass diversity of applications and platforms create an ecosystem that is hard to optimise from every conceivable perspective. The main characters playing a role in this ecosystem are the device manufacturers, application developers, cellular network operators and the users.

While the technology development and the device manufacturers may provide more energy-efficient hardware and batteries in the longer run, the lack of energy awareness on the software front is becoming all too apparent. Even though manufacturers are likely to optimise the built-in software, their devices are shipped to a market where there is no control over the wide range of applications that can be used on them. Most applications are completely oblivious to how their operation and data transmissions affect the energy consumption of the device. Thus we believe that there is still a need for carefully analysing the energy footprint of application software, and to use
software to monitor and reduce the energy consumption.

At the other end, great part of the efforts of cellular operators is directed towards lowering energy consumption in the infrastructure nodes, not especially caring about the user end. This might seem natural since a great part of the operational cost of the network is due to energy consumption (1 billion euro per year [22]), and a typical base station consumes many times more than a mobile handset [16] (more than 1kW for macro, around 800 W for micro, and 14 for pico, whereas a smartphone’s maximum consumption does not reach 5 W).

The consequence of the cellular operator’s capacity optimisation, caring for maximum loads when configuring their radio network settings, becomes a high tax on the handsets’ energy even during underutilisation periods. Cellular networks impose high energy consumption on the mobile device due to the radio resource allocation performed at the operator end. We find that the battery of the users is wasted by the undisciplined use of data transmissions in the cellular network, using applications that were not specifically optimised for a given radio interface or operator settings.

A user that has access to the data and services anytime anywhere will focus on interactions with the applications, expecting to communicate whenever needed without thinking about battery lifetime. However, the above ecosystem leaves the user in an undesirable state. End users have been accepting shorter battery lifetimes in exchange for better connectivity and advanced services for some time. There is a point where the nuisance of charging the phone will outweigh the perceived benefits of increased performance. Battery lifetime has become the least satisfying factor in mobile devices\(^1\), and a decisive factor when selecting the device model or manufacturer. Seen from a user perspective, what counts in the end is that the handset has a long enough battery lifetime.

Last but not least, the aggregated consumption of recharging all the smartphones becomes a significant cost. Given the large number of mobile subscribers (more than 6 billion\(^2\)) and smartphone shipments (1.2 billion by the end of 2013\(^3\)), one can estimate that the average cost of recharging these devices per year is equivalent to the power generated by an small nuclear power plant or 354609 households (see Appendix A for details). As a side effect, extended battery lifetimes would also help to extend the consumer-use phase of a smartphone’s product lifecycle hence reduce the impact of the manufacturing process (e.g. raw-material extraction, component manufacturing) as well as the disposal.


\(^2\)http://mobithinking.com/mobile-marketing-tools/latest-mobile-stats/

\(^3\)http://www.gartner.com/newsroom/id/2227215
1.1 Preliminaries

With the above situation in mind, energy economy in mobile devices has become an important factor, and wireless interfaces account for a great energy cost on mobile devices [38, 78]. There is a need to reduce the energy consumption of wireless data transmissions at the user end and energy awareness is the key starting point to achieve energy efficiency in data communication. Energy agnostic systems and applications are likely to waste energy in inefficient transmissions by not being informed of how their transmissions impact the battery lifetime. Energy-aware design and operation is required in order to achieve energy-efficient systems.

Energy proportionality is a desired feature of every energy-efficient system [20]. This is defined as the relation between system load or utilisation and the energy consumption. Energy proportionality enforces the energy consumption to be low when the utilisation is low, and the energy consumption gradually increases with the utilisation. Energy proportionality is applicable to wireless networking, where the system utilisation can be defined in different ways (e.g., the amount of data transmission or the instantaneous data rate). The application of energy proportionality to wireless networking at the handset end is an area not properly studied so far.

Several aspects impact the energy consumption of a wireless interface, including the hardware chipset of the particular technology (i.e., the set of electronic components in the integrated circuit and its power management) and the transmissions created by the commanding software. The mechanisms employed at lower layers make the energy consumption of data transmission highly dependable on the data traffic pattern and the wireless interface. For example, the power management of 802.11 interfaces (WiFi) differ much from the mechanisms employed for cellular interfaces, and therefore different data patterns consume considerably disparate amount of energy due to the operation of the low layer mechanisms (e.g., a single packet sending in 3G consumes 4.5 Joules, whereas in WiFi consumes 0.04 Joules).

When developing energy-efficient solutions for mobile services, it is important to consider that these exhibit various interaction patterns, and present distinct requirements in terms of Quality of Service (QoS) which limit the solution space. These can present different requirements in terms of timeliness, bandwidth or reliability, which is measured using parameters such as jitter, delay, response time, system level data rate or loss [26]. However, there is so far no requirement from the energy consumption perspective and it is often considered to be part of the Quality of Experience (QoE). While QoS is an objective measure of the service provided to the end user, QoE is often defined as a purely subjective measure representing the user’s perspective of the provided service [45, 107]. In this thesis, the emphasis is on prioritising energy consumption and quantifying it while to some extent ignoring the QoS aspects of the type of application such as interactivity, elasticity, tolerance or adaptivity [33].
When developing energy efficient solutions in order to achieve energy proportionality for wireless interfaces, two general approaches are often devised:

- **Generic solution per traffic class:** Given the characteristics of a traffic class, the energy consumption can be optimised knowing its QoS requirements (if any). The energy consumption of a set of applications generating traffic that falls within the same class can be reduced by a common solution. For example, a traffic characterised for being elastic, best-effort and without any specific QoS requirement (except for the fact that it needs to be sent) introduces flexibility with respect to optimising energy consumption.

- **Tailor-made solution:** Tailor-made energy-efficient solutions are conceivable given the data transmission requirements of a certain type of application. Even though applications may generate different types of traffic, knowledge about the usage of these applications can suggest possible ways of improving their energy efficiency. The application might be willing to balance the inherent performance-energy tradeoff at the design stage or provide some levels of adaptivity. Developers can adopt tailor-made solutions to reduce the energy expenditure while maintaining the particular requirements for some application.

In our work we explore techniques that fall into both of the above categories.

### 1.2 Problem formulation

The purpose of this thesis is to provide mechanisms and tools to address energy consumption problems for wireless network applications. The research goals are the following:

- **Energy-awareness:** Provide efficient means to support the analysis of transmission energy footprint at the user end. The proposed approach should complement physical power measurements, and facilitate the design of energy-efficient mechanisms and network applications.

- **Energy saving techniques:** Develop methods to reduce the energy consumption of cellular transmissions. The goal is to define approaches and mechanisms to achieve energy efficiency for a given traffic class, and develop tailor-made solutions given the application operation knowledge. The proposed approaches should also help to analyse and compare the advantages and disadvantages of tailor-made and traffic class based solutions.
1.3 Contribution

The contributions of this thesis are as follows:

1. A simulation tool to perform transmission energy studies at the device end: We propose EnergyBox, a generic energy simulation tool that captures the influence of the data transmission pattern on transmission energy consumption at the user end, using real data traces as input. The tool provides accurate and repeatable energy consumption studies for 3G and WiFi transmissions. The strength of EnergyBox is that it allows to modularly set the 3G network parameters specified at operator level, the adaptive power save mode for a WiFi device, and the different power levels of the operation states for different handheld devices.

2. Cross-layer background traffic scheduler for 3G: We devise an energy-efficient solution that schedules data transmissions of a particular traffic class at the user end. The elastic and best-effort background traffic is scheduled based on the current network parameters that impact the 3G energy consumption. The means to estimate these parameters and provide them to the scheduler are also developed. The cost of running the solution is quantified with real application data traffic showing low overhead and significant energy savings.

3. Designing energy friendly transmission and solutions for applications: Given the knowledge of different applications energy usage, we propose means to perform energy-efficient data transmissions by developing tailor-made solutions for two type of applications. In particular, we experimentally compare two application layer protocols for location sharing over cellular networks. The evaluation provides guidelines to select which protocol to use given the usage of the application. Moreover, the results show that a more compact data format can significantly reduce the energy consumption. Furthermore, in the context of instant messaging, we quantify the energy cost of a feature commonly employed by applications and show its tremendous energy cost. A mechanism to aggregate messages reducing energy consumption is also proposed.

1.4 List of publications

The work presented in this thesis is based on the following publications:

1.5 Thesis outline

The thesis is organised as follows:

Chapter 2 - Background provides the needed background to understand the rest of the thesis. In particular, the energy consumption problem is explicitly illustrated using real instant messaging applications, the technological background regarding 3G and WiFi is provided, and an overview of the research efforts and directions are described.

Chapter 3 - EnergyBox presents the transmission energy simulation tool for 3G and WiFi. The chapter describes the modelling of the lower layer mechanisms and the validation against physical measurements.
Chapter 4 - Cross-layer Burst-Buffering describes the energy saving technique developed for the background traffic class. The usefulness of EnergyBox to evaluate the energy savings achieved by the algorithm is shown as well.

Chapter 5 - Energy-Efficient Location-Sharing focuses on studying impact of the choice of protocols on the energy consumption of location sharing applications using EnergyBox.

Chapter 6 - Energy-Efficient Instant Messaging describes the work done in the context of instant messaging applications.

Chapter 7 - Related works presents a survey of the related works providing a taxonomy for the main work in the area. The reasons for placing the taxonomy at the end are that Chapter 7 is not fundamental for understanding the thesis, our approaches and others works are more suitably compared once they are described, and the taxonomy includes works that are orthogonal to the main focus of the thesis, but still within the wireless energy efficiency scope.

Chapter 8 - Conclusion and Future Work concludes the work presented in the thesis and describes directions for future work.
Chapter 2

Background

This chapter provides the needed background for an understanding of the contributions presented in this thesis. In section 2.1, the fact that some applications can unjustifiably consume much more energy than similar ones is shown as a reflection of the real world problem. Section 2.2 provides a background for the energy consumption of the technologies (3G and WiFi) used to implement the concepts developed in this thesis. Finally, a brief overview of the research efforts towards solving the energy consumption problem of wireless transmissions at the user end is described in section 2.3.

2.1 Application transmission energy

There are several reasons why different applications can consume different amounts of energy. For example, they might provide different features or implement different communication mechanisms. However, the fact that functionally similar applications can consume completely different amounts of energy is unjustifiable from a user’s QoE perspective. This issue is explored and described through an illustrative example concerning instant messaging applications. The charts in this section are reproduced from the Bachelor thesis by Simon Andersson [11].

We select 6 of the most popular instant messaging applications from the Android market as a test set. Since the applications are products and their code is not available, we consider them as black boxes. The same simple short conversation is created using two different smartphones connected via 3G using the different applications. The 2 minute conversation is shown in Table 2.1. The test is repeated 5 times and we collect the packet traces in the smartphone of the user 1. The energy consumption for each application is calculated using EnergyBox, our tool that is described later in Chapter 3.

Fig. 2.2 shows a great diversity regarding the amount of energy spent by the different applications when exchanging the short conversation. The most consuming application (Messenger) consumes 153% more energy than the...
Table 2.1: Short conversation.

<table>
<thead>
<tr>
<th>Timestamp (min:sec)</th>
<th>User number</th>
<th>Message sent</th>
</tr>
</thead>
<tbody>
<tr>
<td>0:0</td>
<td>user1</td>
<td>Hello</td>
</tr>
<tr>
<td>0:15</td>
<td>user2</td>
<td>Hi!</td>
</tr>
<tr>
<td>0:32</td>
<td>user1</td>
<td>How are you?</td>
</tr>
<tr>
<td>1:01</td>
<td>user2</td>
<td>I am great, thank you!</td>
</tr>
<tr>
<td>1:14</td>
<td>user2</td>
<td>You?</td>
</tr>
<tr>
<td>1:33</td>
<td>user1</td>
<td>I’m good thanks.</td>
</tr>
<tr>
<td>1:48</td>
<td>user1</td>
<td>Bye</td>
</tr>
<tr>
<td>1:56</td>
<td>user2</td>
<td>Bye</td>
</tr>
</tbody>
</table>

Figure 2.1: Transmission energy and standard deviation of different Instant Messaging applications for the short conversation.

least consuming one (GTalk) to transmit the same conversation. This short conversation using Messenger consumes 0.3% of a commonly used mobile phone battery type (2000 mAh at 3.7V). The difference from using the least energy-efficient application could substantially shorten the battery lifetime of a device by a factor of 2.5 reducing the QoE of the user.

The data traffic pattern characterises the transmissions performed by an application. Fig. 2.2 shows the amount of data sent, the number of packets and the empirical cumulative distribution functions (CDF) of packet size and inter-packet interval during the experiments. These parameters are often used to describe the transmission characteristics. For clarity, we select only 3 of the 6 applications for the CDF graphs.

In general, one can observe significant diversity in the different parameters for different applications. WhatsApp and GTalk employ few and small packets, whereas Messenger sends a lot of data and 10% of the packets are greater than 1000 bytes. The high inter-packet interval of GTalk describes sparse data transmissions, and therefore longer idle periods between transmissions than WhatsApp and Messenger. Even though Kik sends much less
data than Messenger, it results in similar energy consumption.

The data encoding, protocol of choice at different layers, added features or functionalities can highly influence the data pattern, and thereby impact the energy consumption. The user interaction becomes a decisive factor when considering the data pattern as well.

![Empirical CDF of packet size and inter-packet interval](image)

Figure 2.2: Data pattern characteristics of the different Instant Messaging applications for the short conversation.

The above measurements have illustrated the need to identify the data pattern of the application (or other system software) and its impact on the energy consumption.

### 2.2 Technological background

This section provides a background for communication energy footprint of 3G and WiFi at the user end through descriptive measurements that show the main factors affecting the data transmission energy.

#### 2.2.1 3G energy consumption

The Radio Network Controller (RNC) is a key element in the UMTS\(^1\) Terrestrial Radio Access Network (UTRAN). It is responsible for the radio resource management and also manages the Node Bs (also known as base

\(^1\)Universal Mobile Telecommunications System
transceiver station), to which the user equipment (UE) connects via radio physical channel. The energy consumption of the UE in 3G is mostly influenced by the Radio Resource Control (RRC) and the Radio Link Control (RLC) protocols, which are defined in the UMTS WCDMA\textsuperscript{2} protocol stack.

According to RRC the UE can be in the states depicted in Fig. 2.3. The states are placed along the y and x axis according to their power consumption and performance in terms of response time and maximum data rate respectively.

![Figure 2.3: 3G connection states.](image)

**RRC States:** In the Dedicated state, a dedicated physical channel (CELL_DCH) is allocated for the terminal in both uplink and downlink providing higher data rates. The terminal has access to dedicated uplink or downlink transport channels, shared transport channels and a combination of them. In the Shared state, i.e., Forward Access Channel (CELL_FACH), the terminal is assigned a default common or shared transport channel in the uplink and monitors the downlink continuously. The UE can transmit small data packets at lower data rates on the Random Access Channel (RACH). While in CELL_DCH and CELL_FACH the UE remains connected to the RNC.

The UE is in the Idle state when there is no network activity. It is not connected but it still can check if there is any downlink packet available. Denoted as Standby in Fig. 2.3, the 3G standard also describes two optional states where the UE maintains a connection to the RNC and the energy consumption is similar to Idle state: the Paging Channel (CELL_PCH) and UTRAN Registration Area Paging Channel (URA_PCH). These two states

\textsuperscript{2}Wideband Code Division Multiple Access
allow the UE to switch faster to higher states. In CELL_PCH no resources are allocated for data transmission and the terminal can use Discontinuous Reception (DRX) to check if there is any downlink packet. DRX reduces the energy consumption by receiving one paging occasion per DRX cycle. The latter is similar to CELL_PCH but it has some performance improvements regarding mobility. Note that some operators do not implement the optional states. In our work, the UE was connected to an operator that implements the URA_PCH state. For the rest of the document we will refer to the different states as DCH, FACH, PCH and Idle.

**State transitions:** State transitions on the UE occur based on traffic volume and inactivity timers controlled by the RNC. Statically set inactivity timers control the state transitions DCH-FACH, FACH-PCH and PCH-Idle, $T_1$, $T_2$ and $T_3$ in Fig. 2.3 respectively. For example, when the UE is in the DCH state for $T_1$ seconds without any or small data transmission, the RNC releases the dedicated channel and switches the UE to FACH by means of the RRC protocol.

The RRC uses information from the RLC protocol [3] in order to report the observed traffic volume to the network. For example, in FACH, the UE reports to the RNC the observed traffic volume based on data buffer status. This helps the RNC to re-evaluate the allocation of resources. The RLC data buffer is used to trigger state transitions. When the content of the data buffer exceeds a certain threshold, the corresponding signalling is performed before switching the state. There are 4 RLC buffer thresholds, 2 uplink and 2 downlink. These buffers are cleared out when the data is transmitted. We will refer to the duration after which the buffers are emptied after transmitting the data as $T_e$. This depends on the data rate of the allocated channel.

![Figure 2.4: Power consumption of different 3G states.](image)

Fig. 2.4 shows the power consumption levels of the states implemented
by TeliaSonera when downloading data. All the physical measurements are performed with our measurement setup described in Appendix B. The 3G module stays in PCH state until there is a transition to DCH, after the dedicated channel connection has been established (3 s in Fig. 2.4). Once the UE is on DCH the data is downloaded. The transition to FACH occurs when all dedicated channels have been released (10 s in Fig. 2.4). The power consumption of in the DCH state is around 1.3 Watts, higher than the FACH state (around 0.5 Watts) and PCH state (0.2 Watts).

It can be noticed that after downloading the data, the UE continues in the DCH state until the inactivity timer $T_1$ has expired causing an energy consumption overhead. A similar overhead is caused in FACH due to $T_2$. The overheads caused by the inactivity timers are called tail effects [18]: $T_1$ leads to DCH_{TAIL} and $T_2$ to FACH_{TAIL}.

Table 2.2: State transitions and triggering packet sizes for TeliaSonera measured in July 2012.

<table>
<thead>
<tr>
<th>State Transition</th>
<th>Uplink threshold (bytes)</th>
<th>Downlink threshold (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCH-DCH</td>
<td>513 - 542</td>
<td>524 - 558</td>
</tr>
<tr>
<td>FACH-DCH</td>
<td>294</td>
<td>515</td>
</tr>
<tr>
<td>PCH-FACH</td>
<td>Always triggered</td>
<td></td>
</tr>
</tbody>
</table>

We measured the uplink and downlink thresholds that trigger the state transitions shown in Table 2.2, by sending UDP packets of different sizes and observing the live power trace using our measurement setup. Note that the PCH-DCH transition is more likely to happen when the triggering packet is closer to the upper bound, i.e., transmitting 513 bytes uplink might not lead to a state transition. Our measurements were carried out over an interval of 5 months, and 3 months into the period the operator settings were changed so that the value of the uplink RLC data buffer threshold that triggers PCH-DCH transition was incremented from the value of Table 2.2 to around 900 bytes (875 - 1000 bytes).

The tail timers and RLC buffer thresholds vary per operator. Fig. 2.5 shows the power profile of operator Hi3g Access AB under the commercial mark 3 measured at our location in Sweden. The variation of the different parameters and the data traffic pattern highly influences energy consumption of the UE, where sporadic transmissions of small packets can lead to high energy consumption. As an illustrative example, we sent a 600 bytes UDP packet every 3 seconds during 1 minute which led to a similar consumption to a Skype voice call (average of 19kB/s): 90 and 92 Joules respectively. This shows that the 3G interface is not energy proportional.

To reduce these energy overheads, the Fast Dormancy (FD) mechanism of the 3GPP Release 8 standard allows the UE to signal the RNC the desire to switch to the lowest power state by sending a Signalling Connection
2.2. TECHNOLOGICAL BACKGROUND

Release Indication before the inactivity timeout. Some networks implement a low activity mechanism in DCH to release the transport channel and move to FACH when there is low traffic [46]. Thus, we see that the energy tails and the above mechanisms make the energy consumption dependent on the traffic pattern and operator settings in a complex way.

To sum up, the statically set inactivity timers and the RLC buffer thresholds at the operator end make the data pattern to drastically impact the energy consumption at the user end.

2.2.2 WiFi energy consumption

The transmission energy consumption for WiFi is mostly influenced at the driver level in the WiFi station (the client handset) depending on the power save mode used. The station is in the Constantly Awake Mode (CAM) when it has the power-saving features disabled experiencing the best performance.

The IEEE 802.11 standard defines a Power Save Mode (PSM), which allows the stations to switch to low power mode during predefined periods of time when not transferring any data. The access point (AP) buffers downlink frames for the clients and the clients wake up periodically (at multiples of the beacon interval). The clients send a Power Save Poll (PS-Poll) message to the AP to receive each buffered frame. Recent smartphones implement a mechanism named Adaptive PSM to overcome the overhead and latency drawback of using this PS-Poll mechanism [58]. The client switches between the CAM and PSM modes based on heuristics (e.g., number of packets, traffic inactivity period or screen on/off). The client uses the power management field in null data frames to inform the AP about its current mode.

Fig. 2.6 shows an adaptive PSM implementation in the Samsung Galaxy SII and Sony Ericsson Xperia Arc smartphones, where the station moves to
CAM for sending the traffic, and switches back from CAM to PSM after a predefined inactivity timeout ($\delta$) without packet transmission. This $\delta$ timeout creates an energy tail in the similar way as in 3G. Repeated measurements have shown $\delta$ to be around 220 and 70 ms for the Samsung Galaxy SII and Xperia Arc respectively, much shorter than in 3G. Previous generation devices implement longer $\delta$ timeouts (e.g., 1.5 seconds for HTC Magic) [67, 91, 102]. Moreover, some drivers also implement packets per second thresholds ($Up$ and $Down$) that trigger PSM-CAM and CAM-PSM transitions respectively [90].

![Power levels of adaptive PSM for Samsung Galaxy SII and Sony Ericsson Xperia Arc.](image1)

Figure 2.6: Power levels of adaptive PSM for Samsung Galaxy SII and Sony Ericsson Xperia Arc.

![Power levels of adaptive PSM for the Sony Ericsson Xperia Arc.](image2)

Figure 2.7: Power levels of adaptive PSM for the Sony Ericsson Xperia Arc.

While in the same state, the station consumes more power when the data rate increases. In order to show the impact of data rate on transmission power and $\delta$, we create an uplink data stream varying inter-packet interval and packet size using the Xperia Arc. Fig. 2.7 shows that for higher data rate...
rates (i.e., short inter-packet interval), the average power level increases substantially. When the inter-packet interval is increased to 70 ms, the station switches back to PSM since the inter-packet interval is greater than \( \delta \). This drops the average power level. To sum up, adaptive PSM leads to a high impact by the data pattern on the energy consumption of a WiFi station.

2.3 Research directions

Saving transmission energy at the user end can be done in several ways. This section provides an overview of the works in the context of transmission energy at the user end. We describe the notions behind the different categories employed to describe the main body of work and their focus. A complete survey of the works that fall within each category can be found in Chapter 7, including the related works. Fig. 2.8 describes the general categorisation of the works, which is described next.

![Figure 2.8: Taxonomy of wireless transmission energy at the user end.](image)

**Measurement studies** typically aim at revealing inefficiencies, discovering and understanding the footprint characteristics of wireless interfaces, and providing guidelines for energy efficient operation. The studies are generally performed by means of physical measurements in mobile devices. The measurement setups employed in these works are typically used for evaluation of energy efficient solution proposals as well.

**Energy saving techniques** are categorised in subgroups based on their aim and the means used to reduce the energy consumption. Our categorisation is inspired by a previous survey of green networking research [21], which categorises the solutions to reduce the energy consumption of networking systems into 4 categories: (1) resource consolidation solutions re-
group all the load from different devices into few of them in order to reduce the global consumption due to underutilised devices at a given time, (2) selective connectedness are distributed mechanisms to allow single pieces of equipment stay idle (e.g., by introducing a low consumption proxy), (3) virtualisation allows more than one service to operate in the same piece of hardware improving hardware utilisation and energy efficiency, and (4) proportional computing encompasses the techniques used to minimise the energy consumption of systems, network protocols, individual devices or components given their utilisation (e.g., dynamic voltage scaling for CPUs).

Our categorisation further refines the taxonomy, by introducing three subtypes for the proportional computing category and a last one for the resource consolidation category:

- **Link selection algorithms** focus on dynamically selecting the best available wireless link at a given point in time based on the fact that different wireless interfaces consume differently. Different technologies and mechanisms are considered to estimate the current network conditions to select the “right” interface. At a given point in time, every link presents a different relation between utilisation and energy consumption. The algorithms select the best link given the utilisation of the system.

- **Low-level power saving** are techniques implemented in the lower layers of the interfaces to make their operation energy proportional. The aim of these works is to improve the operation specified by the standard of particular interfaces (e.g., WiFi PSM), optimising the operational parameters or adapting the operation based on the utilisation knowledge (e.g., high loads or a specific type of traffic).

- **Traffic adaptation** techniques consider the power saving mechanisms as given, and adapt the traffic to reduce the energy consumption. Since the data pattern greatly influences the energy consumption, these techniques attempt to achieve energy efficient patterns by, for example, reducing the amount of data to be sent or batching transmissions. Since the works attempt to improve the energy efficiency given the utilisation of the system, they are categorised as proportional computing works.

- **Cooperation techniques** consider a set of devices whereas previous solutions were focused on a single mobile device. This aim at reducing the aggregated energy consumption of the nodes. Cooperation techniques often employ multiple interfaces in a similar way to link selection techniques. However, the techniques aim at organising the energy expenditure of a group of nodes so that it is minimised. These solutions are categorised as resource consolidation solutions.

The **energy models** works focus on characterising energy consumption for the general operation of wireless interfaces or specific mechanisms. The
models are either analytical or measurement-based depending on the tools used to model the energy. Analytical models are devised to mathematically model specific mechanisms of a wireless interface that impact the energy consumption. Once the models are built, they are used to suggest guidelines to optimise the parameters of the mechanisms. These models require modelled traffic.

Instead, measurement-based models are often generated using real data collected through physical measurements. Their complexity varies greatly and ranges from simple models characterising the energy consumption based on some statistical representation of the measurement data (e.g., linear regression), to more complex models employing finite-state machines. These models can be seen as bottom-up approaches, which can be specific and limited to the measured data depending on the model development approach.

The contributions of this thesis are within measurement studies, traffic adaptation energy saving techniques and measurement-based energy models.
Chapter 3

EnergyBox

The previous chapter has shown how data transmission pattern drastically influences the energy consumption of the end device. The data pattern greatly depends on the real applications operation (e.g., developer choices for application mechanisms, communication protocols or data format) and the user-device interaction [35, 79]. Ergo, the data pattern becomes the prime focus in our work when analysing energy consumption.

While physical power measurements can undoubtedly support the development of energy-efficient solutions for wireless data transmissions, performing power measurements is a non-trivial task which requires some specific knowledge. Designing and setting up tests, performing the actual measurements and analysing the collected data is laborious and time consuming.

There are different approaches available to perform energy studies. However, the different methods expose some limitations considering the goal of performing flexible and efficient energy consumption studies. The professional testing equipment [4] or the physical power measurement equipment [38, 99, 102] (similar to the one described in Appendix B) provide high accuracy, but they limit the studies to a fixed environment and incur high cost. Similarly, vendor specific development platforms [5] provide means to analyse the energy consumption using different current sensors embedded in the different device components. Smart battery interfaces providing current values are also available in some devices, and the software using them (e.g., CurrentWidget for Android or Nokia Energy Profiler) measures the aggregated current draw from all the device components. The error of the instant battery interface reading is usually high [31]. These solutions allow device-dependent studies only, and since the change of hardware between generations can substantially make the energy consumption to differ from previous generation devices, these solutions become obsolete quickly.

We observe that due to the high cost of the measurement solutions and time limitations (e.g., time-to-market of applications), third party software developers cannot afford the invest in these solutions, and applications and
system software are not designed or tested with energy consumption in mind. Therefore, physical measurement are useful to provide insight and observations, but there is a need for tools and solutions that can complement physical power measurements and enable efficient studies to minimise the energy consumption. We believe that a modular approach to carry out flexible and efficient energy studies isolating the transmission energy to complement physical energy measurements is essential.

Our work proposes the design of EnergyBox: a tool that enables accurate studies of data transmission energy consumption at the user end, using real traffic traces as input. We focus our efforts on the most widespread wireless technologies (3G and WiFi) and capture application data transmission energy footprint at the user device. EnergyBox captures the underlying states of operation of the wireless interfaces. The hardware dependence of energy is overcome by using parametrised device specific power levels. For a given data trace, EnergyBox automatically outputs the operation states over time, so that when combined with device specific power levels it enables the computation of energy consumption. The general idea of our EnergyBox is shown in Fig. 3.1.

The data traces can be directly captured on different devices or synthetically created in order to study the impact of different transmission mechanisms under different configurations for these wireless technologies. EnergyBox is intended to assists researchers and application developers to immediately estimate energy consumption of data transmissions for a diverse range of test cases.

The tool accepts the statically configured parameters at the radio layer such as inactivity timers, the data buffer thresholds and a low activity mechanism used by the operator for state transition decisions in 3G communication. For WiFi, it incorporates the adaptive power save mode commonly used as the power saving mechanism in the latest generation devices.

This chapter is structured as follows: section 3.1 describes the design of the 3G and WiFi models included in the EnergyBox. Section 3.2 evaluates the accuracy of the EnergyBox compared to physically measured energy...
CHAPTER 3. ENERGYBOX

consumption. Section 3.3 discusses the results and the current limitations of EnergyBox.

3.1 EnergyBox design

The design of the EnergyBox is rooted in measurements and a careful literature study. EnergyBox performs trace-based iterative packet-driven simulation. The usage of real data traces means that the corresponding energy is realistic, and reflects the impact of the data pattern on real throughput and delay in the network. EnergyBox also accepts using synthetic data traces as input, creating repeatable tests for a given purpose.

Given a packet trace and configuration parameters, the EnergyBox outputs the device states $S(t)$ over time. The total energy consumption is calculated by associating these states with power levels and integrating them over time. Device-specific power level values can be obtained through measurement platforms as the one described in Appendix B or in the literature [18, 91, 99, 114]. These operation power levels abstract the hardware dependency allowing to quantify an application footprint on a given device.

EnergyBox simulates the 3G network parameters specified at operator level and the adaptive PSM mechanism specified at the handset driver for WiFi. We further describe their modelling in the following subsections.

3.1.1 3G model

The RRC state transitions are captured by a parametrised finite state machine that simulate the inactivity timers, the RLC buffers and a low activity mechanism in a packet-driven manner. Fig. 3.2 (a) shows the states and the state transitions we use in our 3G model.

**States:** We define 3 different states in our 3G model: Idle or PCH, FACH and DCH. These are based on the standard RRC state machine. The strength of modelling 3G states in EnergyBox is that it is a generic way of capturing different RRC state machines for different operators using a parametrised finite state machine. If an operator implements a state machine where a single packet triggers a state transition to DCH, we simulate the it by setting the transition parameters according to the state machine. State promotions refer to performing a transition to a higher performance state, whereas state demotion refer to switching down to less performing states.

**State promotion:** For each packet $P_i$ in the trace and its timestamp $t(P_i)$, we calculate $\Delta_i = t(P_i) - t(P_{i-1})$ as the elapsed time between the packet and its predecessor. $\Delta_i$ is used to simulate the inactivity timers $T_1$ and $T_2$: if $\Delta_i > T_1$ or $\Delta_i > T_2$, we trigger the corresponding state transition. In order to account for the signalling time between states, every state transition has pre-defined a parametrised transition duration.
Inactivity timers of different real networks are available in the literature [18, 19, 72, 81, 95] or can be obtained from a single power measurement when sending a single packet (from Fig. 2.4 or Fig. 2.5 in section 2.2.1) and observing the time spent in each state. Similarly, the transition duration times are obtained from a power measurement (e.g., the delay observed in Fig. 2.4) or measuring the round-trip time for the different transitions sending various pings.

Recall that some networks implement a low activity mechanism in DCH to release the transport channel and switch to FACH [46] before the inactivity timer $T_1$ expires. In order to simulate this low activity mechanism, we define $T_d$ as the period of time over which the amount of data sent is monitored. We sum the size of packets during $T_d$ and force the move to FACH when it is lower than a threshold. The Fast Dormancy mechanism that the UE uses to signal the desire to switch to a low power state before the inactivity timeout is modelled by simply moving to PCH/Idle after a predefined time. We represent this behaviour by simply shortening the inactivity timers in case we want to use fast dormancy.

**Figure 3.2**: Overview of the EnergyBox state machines for 3G (a) and WiFi (b).

**State demotion**: There are four RLC buffer thresholds for PCH-FACH and FACH-DCH transitions: two uplink ($B_u^1$ and $B_u^2$) and two downlink ($B_d^1$ and $B_d^2$). The simulation of the RLC buffers is done as follows: we define $\Delta_i^u$ and $\Delta_i^d$ as the elapsed time from the last uplink or downlink packet
(in the same direction). Note that since the size and direction (up/down) for each packet $P_i$ is known, the current occupancy of each buffer in bytes can be computed. We denote this by $C$. The data is transmitted (i.e., the buffer is emptied setting $C = 0$) depending on the channel data rate. Given $C$ and the channel data rate, the time to empty the RLC buffer can be calculated. $T_e$ can be specified as a constant assuming constant data rate (e.g., $T_e = C/512$ kbps) or as function of buffer occupancy $C$ based on data rate measurements in FACH for a network. Similar to the RLC thresholds, $T_e$ also depends on the direction: $T_{ed}$ and $T_{eu}$ for downlink and uplink respectively.

The RLC buffer thresholds from a real network can be acquired experimentally or from the literature. The experimental method used is as follows: For every transition and direction (uplink/downlink), we set the device in the desired state (PCH or FACH), send a packet with the potential triggering size and observe the state transition on live power trace generated by a measurement setup. The size of the packet is increased until the transition takes place resulting in the threshold.

Once the RLC buffer thresholds are known ($B_2$), $T_e$ (uplink and downlink) can be modelled based on data rate measurements instead of assuming a constant data rate [95]. In order to calculate $T_e$, we define a packet of $s$ bytes ($s < B_2$), a packet of size $b$ ($b < B_2 < s + b$ so that the sum triggers the state transition) and a short delay. We send a packet of size $s$, wait the short delay and send $b$. Observing a state transition explains that the time to empty the buffer $T_e$ was greater than the short delay. Since the buffer is not emptied when the second packet arrives to the buffer, the $s + b$ exceeds the threshold triggering the state transition. Then, for every packet size $s$, we shorten the delay until the state transition occurs in order to estimate $T_e$. Fig. 3.3 shows the resulting $T_{eu}$ in the uplink direction for a granularity of 50 ms. The result is different from assuming a constant data rate.

![Figure 3.3: Measured time to empty the buffer $T_{eu}$](image)

**Power levels:** The current model employs fixed power levels for the different states. This is a convenient simplification in order to obtain the power values from a power profile. This assumption is reasonable given that 3G has relatively low data rates and the power levels in the same state are fairly similar. The average power level for the different states is used.
The current 3G model does not consider the impact of the received signal strength (RSS). Under poor radio-link conditions the transmission power is higher and the data rate is lower (as described in 2.2.1). The data rate is captured by the recorded input data trace. However, in order to model the impact of RSS on power level we would have to feed the EnergyBox with a trace of received signal strength or similar indicator (e.g., Signal-to-Noise Ratio). This can be added to the implementation if the means of capturing such traces are available. In the current state of the model, the fixed power values need to be manually adjusted in order to calculate the energy consumption at different RSS.

### 3.1.2 WiFi model

The EnergyBox WiFi captures the adaptive PSM mechanisms based on the inactivity timer and the number of packets per second. Handsets switch between two states (PSM and CAM) using adaptive PSM, but in order to model the high data rate behaviour of a handset, we define the state machine shown in Fig. 3.2 (b).

The station only wakes up for beacons in the **PSM** state. **PSM Transmission** state represents the sending or reception of packets in the PSM mode, where the station switches to a high power only during the transmission interval. The Tx/Rx time is defined as reconfigurable parameter, which is obtained by sending few packets and observing the power profile (similar to Fig. 2.6 in section 2.2.2).

In **PSM**, the transition to **CAM** is triggered whenever the number of packets per second count is higher than the $Up$ threshold. When the number of packets per second is less than $Down$ for a predefined timeout time ($\delta$), the station switches back to **PSM**.

The parameter $Up$ is obtained by increasing the number of packets sent per second and observing the transition to CAM in the power trace. Similar methodology can be used for $Down$ once the station is in the CAM state. The parameter $\delta$ is directly obtained from the power trace or observing the time interval between the NULL frames that the device sends to the access point to describe the state change. These values are also available in the literature for different devices [67, 90].

We employ fixed power levels for the **PSM**, **PSM Transmission** and **CAM** state, which are the average power drain at the state. However, we have observed that the power drain of the station increases with the throughput as it was illustrated in section 2.2.2. This behaviour is captured by the **HighCAM** state. The throughput is computed over a short time window, compared to a $\rho$ threshold, and when the threshold is exceeded, the station switches to **HighCAM**.

Once the time window has been set, the $\rho$ threshold and the power value for **HighCAM** can be decided by sending sequences of packets with different inter-packet intervals (similar to Fig. 2.7 in section 2.2.2) and measuring
the power drain. The \( \rho \) is set to the point in Fig. 2.7 when the power drain significantly increases. The selection of the power level for the HighCAM becomes a tradeoff: a too high power value would lead to overestimation of the consumed energy, whereas a too low value can underestimate it. We empirically select the power value to be between the power level of CAM and the maximum observed power level as a compromise.

### 3.2 Evaluation

This section describes the evaluation of the EnergyBox accuracy against physical energy measurements. We define two metrics to quantify the accuracy of the EnergyBox: energy accuracy and time accuracy, which are described together with the methodology. The general methodology is similar for both WiFi and 3G and follows the following steps: (1) A set of applications are used to create different traffic patterns. We simultaneously collect packet traces and measure the device power trace \( P_d(t) \) creating traffic from one application at a time. This is done using the toolset described in Fig. B.1. The power trace \( P_d(t) \) represents the ground truth. (2) From the measured power trace, we compute the expected device states \( S_d(t) \) using power level thresholds. These thresholds are obtained earlier using measurements on a given handset. (3) The EnergyBox is fed with the packet traces and outputs the inferred states \( S_i(t) \). (4) We compare the EnergyBox inferred states \( S_i(t) \) against the device states \( S_d(t) \) and compute the difference over time. \( T \) represents the total duration of a trace and \( f(x, y) \) simply returns 1 if \( x = y \), and 0 otherwise. Time accuracy represents the percentage of time that the inferred states and the measured states overlap (i.e. the higher the measure the better the accuracy over a time interval). It is defined as follows:

\[
\text{Time Accuracy} = \frac{\int_0^T f(S_i(t), S_d(t)) \, dt}{T} \tag{3.1}
\]

(5) The EnergyBox assigns the measured device-specific handset power levels for each state to the inferred states obtaining the inferred power trace \( P_i(t) \). The energy consumption is computed in Joules integrating \( P_i(t) \) over the trace duration (\( T \)). Energy accuracy reflects the difference between the inferred energy consumption and the energy consumption of the measured power trace (the ground truth):

\[
\text{Energy Accuracy} = \frac{\int_0^T P_i(t) \, dt}{\int_0^T P_d(t) \, dt} \tag{3.2}
\]

### 3.2.1 Dataset and evaluation settings

The packet traces are captured with \textit{tcpdump} for WiFi and 3G. We demonstrate the reliability of the EnergyBox by covering a wide range of data patterns in terms of inter-packet interval, packet size and total amount of
data transmitted created by commonly used mobile applications. Since the EnergyBox is deterministic (i.e., it creates the same output for a given input packet trace and the configuration parameters), each packet trace is fed only once in the EnergyBox.

We chose to test several different applications with one trace each, similar to the approach in other current work [31, 78, 94]. We employ 10 different packet traces coming from different applications for 3G and 9 for WiFi. They are 5 minutes long. Email has periodic small data transmissions, whereas Facebook and Web represent bursty downloads of bigger amounts of data. Spotify is a music streaming application that sends data in different bursts and Stream is a constant radio stream. Skype Chat represents instant messaging services which usually have smaller data transmissions. Skype Call and Video are audio and video conferences with some small chat messages. Finally, Youtube captures the user watching videos online.

For 3G, we set $T_1 = 4.1$ s and $T_2 = 5.6$ s and the following RLC buffer thresholds: $B_{u1} = 1000$, $B_{d1} = 515$, $B_{u2} = 294$ and $B_{d2} = 515$. The state transition times are 1.7, 0.43, 0.65 s for PCH-DCH, PCH-FACH and FACH-DCH respectively. We set the time to empty the RLC buffers, with $u$ and $d$ denoting uplink and downlink respectively. More specifically, $T_{e}^{u} = 1.2C + 10$ as a linear function of the buffer occupancy ($C$) based on measurements. This is done by measuring the time interval between packets that trigger the state transition in a series of empirical tests in the given network. The $T_{d}$ parameter is set as a constant to 30 ms. We enable the low activity mechanism (with $T_d = 4$ s and $B_{u1}$ as the threshold) and disable fast dormancy since the 3G module in the measurement kit used in the experiments does not support it. The device-specific power levels used are the average power values measured on the 3G module for different states: DCH, FACH and PCH (1.3, 0.5 and 0.2 W). The evaluation was performed under similar and typical values of received signal strength at a single location.

For WiFi, we set $Up = 1$ and $Down = 1$, which are the settings of the stations used in our measurements. The Galaxy SII was used for the evaluation and $\delta$ was set to 220 ms based on observation (similar to Fig. 2.6). We used 30, 250 and 500 mW for PSM, CAM and HighCAM respectively based on measured average power levels. The $\rho$ data rate threshold was empirically set to 3kB for a time window of 50 ms based on empirical experiments mentioned in section 2.2.2. Since adaptive PSM in the Galaxy SII differentiates only two states (PSM and CAM), we perform the time accuracy evaluation of the EnergyBox WiFi using the two basic states of our WiFi state machine.

### 3.2.2 Accuracy of EnergyBox 3G

The average accuracy range of EnergyBox over the different traces for 3G is high, 95% and 98% for time and energy respectively. Fig. 3.4 shows the time and energy accuracy for the different traces starting at 80%.
Regarding lower time accuracies (e.g., Skype Chat), the typical cause is that the inferred state is DCH state while the real state is FACH. The source of this deviation is the time to empty the buffers ($T_e$) of the RLC buffer simulation. Even though the modelling of this parameter is based on real measurements, the variations in number of active users and traffic at the operator network makes the real time to empty the buffers different from the measurement-based model in some instances. The Skype Chat trace has many small packets. These stress the RLC data buffer simulation. However, the accuracy is still close to 90%.

The energy accuracy is high in all the traces. The lowest registered energy accuracy is for Email and Spotify traces, which is still higher than 94%. Fig. 3.5 shows an example of the EnergyBox accuracy by comparing it to the real measurements for the Email trace.
3.2. EVALUATION

Fig. 3.6 shows an example of the source of inaccuracies for the Web1 trace. The simplicity of the fixed power level for the different states is shown compared to the moments when the data rate is high. However, the impact on the error is low.

![Figure 3.6: A fragment of a 3G measurement and EnergyBox 3G inferred output for the Web1 trace.](image)

3.2.3 Accuracy of EnergyBox WiFi

The average accuracy of EnergyBox over the different traces for WiFi is really high with 99% and 98% for time and energy respectively. Fig. 3.7 shows the accuracy for the different traces. Concerning time accuracy, the few discrepancies are originated from the value given to \( \delta \), which makes the switch from CAM to PSM somewhat earlier than in the real states. Regarding the energy accuracy metric, the differences can be traced to traces with higher data rates (e.g., Skype Video and Stream). The simplicity of the fixed power level for *HighCAM* in our model does not completely cover the
case of high data rate with a higher power level than the fixed one assigned to HighCAM. However, the discrepancy is still relatively small.

To summarise, the above comparisons provide evidence that EnergyBox provides high energy accuracy and device state information when using real traces.

3.3 Limitations and discussion

Designing energy efficient solutions for wireless networks starts by energy awareness. To reduce energy waste, there is a need to provide visibility and understanding of the energy consumption that impacts battery lifetime at the user end as well as elsewhere in the network. Our contribution makes the energy footprint of the network interface explicit, and presents a tool that enables efficient, flexible and accurate energy consumption studies given packet traces as input. The users of the tool can easily quantify the energy consumption of their transmissions for provided scenario setups.

We build on the fact that the handset energy consumption in 3G is essentially driven by the inactivity timers and the RLC data buffer thresholds that control RRC state transitions at the network operator end. The energy consumption in WiFi is mostly influenced by the handset dependent adaptive PSM mechanism implemented at driver level in the handset. We capture these aspects within parametrised state machines in the EnergyBox and evaluate its accuracy against physical measurements. Our evaluation shows that the EnergyBox provides accurate energy consumption estimates.

EnergyBox has currently two modelling limitations. The fixed power level for the different states is a convenient simplification which also leads to inaccuracies in high data rate cases, especially in the WiFi case. Adding dynamic power levels to the most consuming states could improve the accuracy. This could be achieved by performing an experiment increasing the data rate and collecting the resulting power. However, it would be device specific.

The data pattern drastically influences the energy consumption, and since the aim of EnergyBox is to quantify its impact on the transmission energy, including the impact of the signal strength was left as an extension. Including the impact of the signal strength into EnergyBox would enable to study the impact of dynamically changing location while transmitting the data. However, this incurs in some complexity for the user of the tool at the time of designing the scenario. The user would need to provide an signal strength trace (e.g., RSS) as well as the packet trace as input.
Chapter 4

Energy-Aware Cross-Layer Burst-buffering

This chapter proposes an online background traffic scheduling algorithm that trades transmission delay for energy consumption by being aware of the transmission energy characteristics of cellular networks. EnergyBox is shown to be a convenient tool to perform the evaluation of energy savings.

Most applications are unaware of the transmission energy characteristics of cellular networks, where the energy consumption is not proportional to the amount of data sent. This unawareness often results in applications performing undisciplined transmissions even without user interaction. In particular, the background traffic created by applications (e.g., periodic transfers or updates) leads to high energy consumption, even though its contribution to the total traffic is not.

As an illustrative example of the energy consumption impact of the background traffic on the battery depletion, 10 minutes of active background traffic (no user interaction or use of the device and the screen-off) can consume 1.6% of the energy of a fully charged average battery (2000 mAh at 3.7 V) without the user noticing it. According to data traffic collected from 20 volunteers [48], 35% of the total traffic was categorised as screen-off traffic which consumed 58% of the total radio energy consumption.

Since applications create traffic with different requirements (e.g., delay sensitive traffic or best-effort), approaches to potentially reduce the energy consumption need to consider these requirements. In this chapter we focus on developing a generic solution that attempts to reduce the energy consumption of the elastic and best-effort background traffic created by applications, which does not have any strict QoS requirements. These traffic is often categorised within the background traffic class specified by the QoS of the third generation UMTS standard. The UMTS background traffic class characterised by low bandwidth requirement, intermittent and asymmetric (uplink/downlink) data transmissions and a permissible delay since
the destination does not expect the data within a certain time. The lack of strict timing or performance requirements enables the different potential energy-efficient approaches to explore the performance-energy tradeoff.

The chapter is organised as follows: We first motivate the need of knowledge about the radio layer parameters to perform energy-efficient background traffic transmissions in cellular networks in section 4.1. Since these parameters are not available for application layer, we propose algorithms to infer them in section 4.2. The algorithm is described in section 4.3 and evaluated in sections 4.4 to 4.7.

4.1 Motivation

Our approach is motivated in the fact that the impact of network parameters such as inactivity timers, RLC data buffers thresholds or quality of radio links on the energy consumption of the UE is high. Figure 4.1 shows an example where small data packets from a data trace are aggregated and sent in a burst causing a state transition to DCH, and higher energy consumption than sending them in FACH. In the original uplink data traffic 7 packets of 100 bytes are sent every 300 milliseconds. When these packets are aggregated to send in burst they lead to 42% higher energy consumption. In order to schedule the background traffic in an energy-efficient manner, the knowledge of the radio layer parameters becomes essential.

Previous algorithms [18,65,101,125] perform data aggregation and burst transmissions because they are not aware of radio layer parameters such as RLC data buffer thresholds. Obviously this is not optimal in the sense that these approaches will aggregate small data transfers causing state transitions to the most consuming state when unnecessary.

Since network parameters are not available for the application layers, we developed algorithms to provide estimates of the state transition timers and buffer thresholds to our scheduling algorithm.

4.2 Parameter inference algorithms

In this section the algorithms to infer the inactivity timers and the RLC data buffer thresholds are presented. These are later on used by the background traffic shaping algorithm. While these algorithms are focused on 3G networks, a similar procedure could be employed for other types of networks.

4.2.1 Inactivity timer estimation algorithms

The algorithms to infer $T_1$ and $T_2$ (inactivity timers from the RRC state machine) are based on the knowledge that each state has qualitatively different round-trip time (RTT) for a data packet. The estimation of the inactivity timers can take place whenever a change of the network parameters is
suspected. We call this a *ReconfigurationEvent*. To calculate RTT, we send synthetic test packets with predetermined sizes from the UE. A server echoes the answer allowing the computation of RTT. We adopt two different approaches for estimating $T_1$ and $T_2$. In both estimations the UE starts in the PCH state. Before presenting the algorithm in detail (Algorithm 1), we provide an overview below.

**Overview:** For the DCH timer $T_1$, we trigger a state transition to DCH by sending a large test packet followed by a sequence of small packets.
At some point, the UE will switch down to FACH state due to the small packet sizes. At this time, the longer RTT will be the indicator of the state transition. Using this process we also have an indication of the RTT in the DCH and FACH states (DCH-RTT and FACH-RTT).

For the FACH timer $T_2$, the above regime would not work. Sending small packets would keep the UE in FACH indefinitely. Therefore, we begin by a guess for lower and upper bound of $T_2$. Note that the upper and lower bound need not be accurate values. We can start by any guess and iteratively converge to $T_2$ by reducing the gap between them. We start with a trial timer value between the bounds. We send a test packet that triggers FACH, wait for the trial timer and adjust the lower and upper bounds based on the UE state. If UE is in PCH, the chosen trial timer is longer than $T_2$. If UE is in FACH, the trial timer is shorter than $T_2$. Using binary search and a sequence of send-wait-checks we converge to an accurate value of $T_2$.

Algorithm 1 describes the two estimation procedures.

**Input variables:** InterPacketInterval and TestPeriod are used for inferring the value of $T_1$. InterPacketInterval defines the time between the sending of test packets and TestPeriod is the duration of estimating the value of $T_1$. Note that this period has to be longer than the actual (unknown) $T_1$. As described above, LowerBound and UpperBound are used for the binary search of the value of $T_2$ until the gap between them is smaller than MinGap. Note that UpperBound should be longer than the actual (unknown) $T_2$. All values are specified in seconds.

**Algorithm description:** We further describe the part of Algorithm 1 that infers $T_1$ (lines 2-15) using an example. We define as large test packet a packet that will trigger the PCH-DCH state transition (e.g., 1200 bytes). A small test packet is the minimum size packet. Figure 4.2 shows the RTT values computed by running Lines 3-12. In line 13, the DetectRTTJump procedure simply identifies an RTT increase in orders of magnitude. In the example, the increase shown for packet number 9 indicates a DCH-FACH transition. Therefore, the procedure returns $j$ equal to 9, which leads to the value of $T_1$ to be 4.5 seconds (9 times 0.5).

Lines 16-29 infer the inactivity timer $T_2$ by performing the aforementioned binary search. In line 17 the UE sends a small packet that triggers the transition to FACH state. The UE waits trialTimer seconds and checks its state by computing the RTT. If the state is still FACH, $T_2$ is longer than the trial time and the lower bound is updated. Otherwise, if the state of the UE is PCH, $T_2$ is shorter than the trial time and the upper bound is updated. This process continues until the gap between the lower and upper bound is smaller than MinGap.

### 4.2.2 RLC data buffer threshold estimation

We next describe the algorithm for inferring the RLC data buffer thresholds that trigger state transitions. There are typically four thresholds, two for
4.2. PARAMETER INFERENCER ALGORITHMS

Algorithm 1 Inactivity timer inference

**Input:** InterPacketInterval, TestPeriod, LowerBound,
UpperBound, MinGap

**Output:** $T_1, T_2$  //Inferred parameters

1. for each ReconfigurationEvent do
   // $T_1$ inference. UE starts in PCH.
   2. $n \leftarrow \text{TestPeriod}/\text{InterPacketInterval}$
   3. for $i = 1$ to $n$ do
      4. if $i = 1$ then
         5. send large test packet  //Force move to DCH
      6. else
         7. send small test packet
      8. end if
      9. receive echo test packet
     10. compute RTT($i$)
     11. wait InterPacketInterval
     12. end for
     13. $j \leftarrow \text{DetectRTTJump(RTT)}$
     14. $T_1 \leftarrow \text{InterPacketInterval} \cdot j$
     15. FACH-RTT $\leftarrow \text{average(RTT}(j + 1), \text{RTT}(n))$
        // $T_2$ inference. UE starts in PCH.
     16. while $(\text{UpperBound} - \text{LowerBound} > \text{MinGap})$ do
         17. send small test packet
         18. trialTimer $\leftarrow (\text{UpperBound} - \text{LowerBound})/2$
         19. wait trialTimer
         20. send small test packet
         21. receive echo test packet
         22. compute RTT
         23. if RTT is close to FACH-RTT then
             24. LowerBound $\leftarrow$ trialTime
         25. else
             26. UpperBound $\leftarrow$ trialTime
         27. end if
         28. end while
     29. $T_2 \leftarrow$ trialTime
     30. end for
the PCH-DCH transition ($B_1^u$ and $B_1^d$) and two for FACH-DCH transition ($B_2^u$ and $B_2^d$) for uplink and downlink. Algorithm 2 describes our approach for inferring the PCH-DCH and FACH-DCH thresholds respectively. As in the previous algorithm, we use the RTT to infer the state of the UE by sending a small test packet and receiving the echo from the server.

For inferring the PCH-DCH data buffer threshold, we use an estimate of the buffer size denoted by an upper and lower bound ($MaxBytes$ and $MinBytes$) in an analogous manner to Algorithm 1. $T_1$ and $T_2$, inferred by the previous algorithm, are used to define a boundary for every trial starting from PCH. This is shown in lines 1-17 of Algorithm 2.

For inferring the FACH-DCH data buffer threshold, we adopt a different approach. In this case, by triggering fewer state transitions we save energy used up for inference. This part of the algorithm works in two phases. In phase one, we approach the buffer threshold using large increments to the test packet size. In the second phase we use small increments in order to fine-tune the estimate of the threshold.

We illustrate the second part of the part of Algorithm 2 that infers the FACH-DCH buffer threshold in Fig. 4.3. The impulses show the time and the size of the sent trial test packets (the packets to compute the RTT are omitted). The test packet size is increased in FACH by a large increment (50 bytes) until the DCH state is triggered. The 300 bytes sent in time point 5 triggers the state transition, therefore the buffer threshold is between 250 and 300 bytes. The RTT calculated with each packet is used to infer the current UE state. Then, after waiting $T_1$ to return to FACH from DCH in time point 10, the size increment is set to $SmallIncrement$ and the algorithm starts sending 250 + increment bytes again (e.g., 260 bytes in time point 10). Note that the $InterPacketInterval$ has to be large enough to allow the clearing of the RLC data buffer. The example in Fig. 4.3 uses an $InterPacketInterval$ of 1 second.
Algorithm 2 Buffer threshold inference

Input: \( T_1, T_2, \text{FACH-RTT} \)

//parameters for PCH-DCH
MaxBytes, MinBytes, MinDiff,
//parameters for FACH-DCH
InterPacketInterval, LargeIncrement, SmallIncrement

Output: \( B_1, B_2 \) //Buffer thresholds in a direction (uplink/downlink)

1: for each ReconfigurationEvent do
//PCH-DCH buffer threshold inference. UE in PCH.
2: while (MaxBytes - MinBytes > MinDiff) do
3: trialSize ← (MaxBytes - MinBytes) / 2
4: send trialSize test packet
5: wait InterPacketInterval
6: ComputeRTT()
7: if RTT is close to FACH-RTT then
8: MinBytes ← trialSize
9: wait \( T_2 \)
10: else
11: MaxBytes ← trialSize
12: wait \( T_1 + T_2 \)
13: end if
14: end while
15: \( B_1 ← \) trialSize
//FACH-DCH buffer threshold inference. UE in FACH.
16: RTT ← \text{FACH-RTT}
17: trialSize ← 0
18: increment ← LargeIncrement
19: for phase = 1 to 2 do
20: while RTT is close to FACH-RTT do
21: trialSize ← trialSize + increment
22: send trialSize test packet
23: wait InterPacketInterval
24: ComputeRTT()
25: wait InterPacketInterval
26: end while
27: increment ← SmallIncrement
28: trialSize ← trialSize - LargeIncrement
29: wait \( T_1 \)
30: end for
31: \( B_2 ← \) trialSize + LargeIncrement
32: end for
4.2.3 Evaluation of inference algorithms

In this section we experimentally validate the algorithms to infer the inactivity timers and the buffer thresholds, and evaluate their energy cost and elapsed time to complete. We are interested in the energy cost in order to later understand whether the cost of inferring the parameters drastically reduces the energy savings or not.

We implemented Algorithms 1 and 2 in Java using UDP packets to calculate the RTT. A simple UDP server was developed to echo UDP packets. The algorithm to infer the buffer thresholds (Algorithm 2) was also implemented in the UDP server to infer the downlink thresholds. The energy consumption of the algorithm was evaluated using the broadband module described earlier.

Algorithm 1: We compared the values inferred by the algorithm to the time that the UE stays in DCH and FACH (observed in the recorded power trace) when triggering a PCH-DCH transition. The settings used in the implementation of Algorithm 1 were the following: small and large tests packets were set to 43 and 1200 bytes respectively (1 byte UDP packet and an arbitrarily large one). InterPacketInterval was 0.5 seconds, providing the $T_1$ with a granularity of 0.5 seconds. InitialGuess, LowerBound, UpperBound and MinGap were set to 7, 0, 16 and 0.5 seconds for the tests respectively. The UpperBound needs to be larger than the sum of $T_1$ and $T_2$, and the InitialGuess is a value in between.

We also compared our algorithm to the one presented by Qian et al. [95] in terms of energy consumption and elapsed time. In short, the idea behind their state demotion inference algorithm is to use the RTT difference between states to infer the state transitions and therefore inactivity timers. Their algorithm sends a packet that triggers PCH-DCH state transition and
4.3. CROSS-LAYER BURST BUFFERING ALGORITHM

The Cross-Layer Burst Buffering (CLBB) algorithm builds on the concepts of proportional computing. Given that cellular interfaces have different per-
forming states, the aim is consume low energy consumption by employing
the least consuming states when the utilisation is low (i.e., map low utilisation
to low energy consumption).

Regarding 3G, this translates to minimise energy consumption by means
of scheduling small data packets in FACH state and avoiding costly state
transitions to DCH that lead to the energy overheads due to $T_1$ and $T_2$
inactivity timers. In addition, we will stop sending packets before their
latest time to send while we are in the PCH state to extend idle time. Our
algorithm is divided in two parts: a packet driven mechanism that inserts
application packets in two different queues depending on their size, and a
UE state based mechanism that transmits the actual packets.

Let us define the problem as follows. Let $A_f = \{a_1, \ldots, a_n\}$ be the set of
running foreground applications with strict time constraints (c.f. hard real-
time) that perform data transmissions (e.g., user interaction). Let $T_f$ be the
shortest relative deadline for each packet generated by an application in the
set $A_f$. Let $A_b = \{b_1, \ldots, b_n\}$ be the set of applications that generate traffic
with no time constraints (background traffic). We define $T_w$ as maximum
waiting time for a packet transmission originated by $A_b$, which is the required
time metric to perform the data transmission (note that all applications have
such maximum time since otherwise one would just block the traffic). In
practice, $T_f < T_w$.

We further classify the packets of the set of applications ($A_f \cup A_b$) in
two categories according to their size L (for large) and S (for short). Let $Q_S$
and $Q_L$ be queues of packets queued according to their size and sorted by
inserting the shortest time to send at the front of the queue. Algorithm 1 and
2 infer the inactivity timers $T_1$ and $T_2$, and the RLC buffer thresholds $B_1$
and $B_2$ (state transition from PCH to DCH and FACH to DCH respectively).
$T_e$ is the time to empty the RLC buffers in FACH which depends on the
current data rate of the channel.

Let us elaborate on the time requirements for the packets. $T_w$ represents
the time requirement of the background traffic, e.g., updates from RSS, e-
mail or Facebook applications. These can be set by the applications or based
on user energy requirements (energy profiles). $T_f$ is defined as a deadline
for other applications. Our algorithm only schedules the background data
transmissions, i.e., the packets with a maximum waiting time and no strict
time requirements. $T_w$ is in practice the maximum time that the algorithm
can delay a packet transmission from $p.t$, i.e., the time that the packet was
originated at the application.

Algorithm 3 describes the energy-aware cross-layer burst buffering algo-

Algorithm 3 (lines 1-8) simply queues the packets depending on their size sorted by shortest time to be sent ($t$) after delaying their latest time to send by $T_w$. $Q_S$ contains packets that can be sent in FACH without triggering a state transition to DCH.

The main part of Algorithm 3 (lines 9-48) transmits data based on the
UE state if there is any packet to be sent. The actions that the UE can
Algorithm 3 Cross-layer burst buffering

Input: $T_w, B_1, B_2, T_e$

1: for each new packet $p$ do
2: \quad $p.t = p.t + T_w$
3: \quad if $p.size \geq B_2$ then
4: \quad \quad $Q_L \leftarrow p$ sorted by $t$
5: \quad else
6: \quad \quad $Q_S \leftarrow p$ sorted by $t$
7: \quad end if
8: end for

//UE state-based packet scheduler. UE in PCH.
9: while $Q_L \cup Q_S \neq \emptyset$ do
10: \quad if UE state = DCH then
11: \quad \quad transmit $Q_L$
12: \quad \quad transmit $Q_S$
13: \quad end if
14: \quad if UE state = FACH then
15: \quad \quad sum $\leftarrow 0$
16: \quad \quad while $Q_S \neq \emptyset$ do
17: \quad \quad \quad $p \leftarrow Q_S.front$
18: \quad \quad \quad sum $\leftarrow p.size + sum$
19: \quad \quad \quad if sum $> B_2$ then
20: \quad \quad \quad \quad wait $T_e$
21: \quad \quad \quad \quad sum $\leftarrow p.size$
22: \quad \quad \quad end if
23: \quad \quad end while
24: \quad end if
25: \quad if UE state = PCH then
26: \quad \quad $T_{nearest} \leftarrow$ shortest $p.t \in (Q_L \cup Q_S)$
27: \quad \quad wait until $T_{nearest}$
28: \quad \quad choose $p \in Q_L \cup Q_S$ with $T_{nearest}$
29: \quad \quad if $p \in Q_L$ then
30: \quad \quad \quad transmit $Q_L$ \hspace{1cm} //Leads to DCH
31: \quad \quad \quad else //Packet with shortest time to transmit is in $Q_S$
32: \quad \quad \quad \quad compute $V_L \leftarrow \sum p.size \forall p \in Q_L$
33: \quad \quad \quad \quad if $V_L \geq B_1$ then
34: \quad \quad \quad \quad \quad transmit $Q_L$ \hspace{1cm} //Leads to DCH
35: \quad \quad \quad \quad else //Choose the channel based on $Q_S$
36: \quad \quad \quad \quad \quad compute $V_S \leftarrow \sum p.size \forall p \in Q_S$
37: \quad \quad \quad \quad \quad ch $\leftarrow$ ChannelChoice($V_S$) \hspace{1cm} //Function 1
38: \quad \quad \quad \quad \quad if ch = DCH then
39: \quad \quad \quad \quad \quad \quad transmit $Q_S$ \hspace{1cm} //Leads to DCH
40: \quad \quad \quad \quad \quad \quad transmit $Q_L$
41: \quad \quad \quad \quad \quad else
42: \quad \quad \quad \quad \quad \quad transmit $Q_L$ \hspace{1cm} //Leads to FACH
43: \quad \quad \quad \quad \quad \quad wait $T_e$
44: \quad \quad \quad \quad end if
45: \quad \quad \quad end if
46: \quad \quad end if
47: \quad end if
48: \quad end if
49: end while
**Function 1 ChannelChoice**

**Input:** $V_S$ //Volume of $Q_S$ in bytes  
$P_1, P_2$ //Power consumption of DCH and FACH  
$T_1, T_e, B_2$

**Output:** channel

```plaintext
if $V_S < P_1/P_2 \cdot B_1 \cdot T_1/T_e$ then
    channel ← FACH
else
    channel ← DCH
end if
```

The UE starts from the PCH state and for every packet sending is able to distinguish between a state transition to DCH or FACH with the knowledge of the RLC buffer thresholds and and the inactivity timers $T_1$ and $T_2$. Every time a packet is sent it updates the time of the last transmission and the current state. After the inactivity timer duration the state of the UE is changed.

When the UE is in DCH we transmit the packets stored in $Q_L$ and $Q_S$. When the UE is FACH, it transmits the packets stored in $Q_S$ avoiding to trigger a costly state transition to DCH, i.e., at limited data rate. This is done by sending a number of packets that sum up to less bytes than $B_2$. Then, the UE waits $T_e$ before the next sending of packets. This allows the UE to remain in FACH state until $Q_S$ is emptied.

Finally, when the UE is in PCH, the algorithm compares the time to send $t$ of packets in the front of $Q_L$ and $Q_S$ and waits for the nearest time to send $T_{nearest}$. Next, the UE decides if it will send a packet or burst that triggers a state transition to DCH or FACH. There are three cases.

First, in case the packet with the shortest time to send is in $Q_L$ (line 29), it needs to be sent. $Q_L$ will be transmitted triggering a transition to DCH. Second, if the packet is in $Q_S$, the UE will transmit $Q_L$ if the volume in bytes of $Q_L$ is larger than the PCH-DCH buffer threshold (line 31-34). This will lead to a trigger of a transition to DCH.

In the third case, the transmission in FACH is feasible with the current volume of the queues. However, a large volume of $Q_S$ can lead the UE to stay in FACH for a long period, which would consume more energy than sending the queues in burst in DCH. The choice of the less consuming option, i.e., sending on DCH or FACH, is performed by Function 1 ($ChannelChoice$) in line 37 given the volume of $Q_S$. If sending in FACH is less consuming, the UE will transmit $Q_L$ which triggers a FACH transition and wait $T_e$. $Q_S$ will be emptied in the FACH state. If sending in DCH is less consuming, the UE transmits both queues in burst triggering a DCH transition.

Function 1 is an optimisation to avoid the case when a large volume of data in $Q_S$ ($V_S$) leads the UE to stay in FACH for a long period. Given the power consumption of DCH and FACH ($P_1$ and $P_2$) and the transmission...
time ($\epsilon$) of $Q_S$ in DCH, the energy cost of transmitting in DCH is estimated by $E_1 = P_1(T_1+\epsilon)+P_2 T_2$. Given the high data rate of DCH, we consider that $\epsilon$ is negligible for the estimation. The energy cost in FACH is estimated by $E_2 = P_2(T_2+T_e \cdot V_S / B_2)$. Therefore, when the inequality $E_1 > E_2$ holds, ChannelChoice will return FACH, otherwise DCH.

4.4 Evaluation methodology

In order to evaluate the total energy savings, we are interested in calculating the networking energy savings that the algorithm provides, as well as the cost of running the algorithm.

The broadband module development kit used to measure the 3G transmission energy (Appendix B) provides the most accurate evaluating method to quantify the networking energy savings by isolating transmission energy. However, we cannot measure the overhead of running the algorithm using the development kit, and thus the algorithm was implemented in a commodity device to enable the evaluation of the energy cost of running it.

The evaluation of the cross-layer burst buffering algorithm is performed in two different environments. The implementation of the algorithm in a handheld device provides the needed data to compute the total energy savings (networking energy savings minus the cost of running the algorithm). EnergyBox is conveniently employed to calculate the networking energy savings while a simple CPU model is derived to compute the CPU cost of running the algorithm. The second environment is used to compare the networking energy savings of the algorithm against the state-of-the-art performing physical measurements in the broadband module.

We begin this section by briefly describing the implementation of the algorithm in 4.4.1 and present the two evaluation environments and methodologies in sections 4.4.2 and 4.4.3. The evaluation results obtained with the first evaluation environment are presented in section 4.5. Then, section 4.6 presents the results of the networking energy savings of our algorithm measured in a broadband module. Finally, as an exploratory phase, section 4.7 describes our experiments when using our algorithm implementation with live traffic.

4.4.1 Algorithm implementation

This section provides the details of an Android implementation of the CLBB algorithm as a Kernel Level Shaper (KLS) [103] in the Android platform. The implementation was developed by Joseba Sanjuan as part of his Master Thesis work supervised by the author.

The architecture of the KLS resides in kernel space since the throughput in kernel space is higher [62], and a user space application would require to interact with the kernel space through system calls. This would create a higher CPU footprint due to data transfer (read/write), e.g., a single data
transfer can result in a CPU usage up to 33% utilisation [47]. The implementation creates a Netfilter Kernel Loadable Module (LKM) that can be inserted at runtime, instead of changing the native code at kernel compiling time. Netfilter is a Linux kernel framework that enables packet interception and modification at different hooks (decision points in the networking stack). The implementation registers the Netfilter NF_IP_POST_ROUTING hook in the IP layer. This allows us to intercept the outgoing packets without interacting with applications, leading to low CPU footprint. The architecture is deployed in a rooted Android device (HTC Sensation Z2710e). We cross-compile the Linux kernel with the needed options (e.g., enable module loading support and Netfilter), boot the new kernel in the device, cross-compile our KLM and load it at runtime.

### 4.4.2 Energy assessment methodology

This section starts by describing the data generation approach in order to evaluate the implementation of the algorithm. Then the methods to compute networking energy consumption and CPU consumption are described.

**Data generation:** In order to provide experiment repeatability and control over the traffic generation of applications, real background packet traces were captured and replayed with and without the KLS in the Android device. We call this emulated traffic since we replay previously captured real packet traces. The choice of the emulated traffic is motivated by the need to create the same application traffic in various experiments to calculate the networking energy savings. These traces contain best-effort traffic that have no specific QoS requirements.

For the emulated traffic, `tcpdump` and an `iptables` based firewall were used to capture only the traffic from the chosen applications. Facebook, Skype and WhatsApp were used as test applications and all the traces were recorded with the screen switched off and without any user interaction. These applications created different traffic load over time based on their operation. Since the energy saving of the algorithm depends on the amount of traffic, out of the used traces we selected what we believe are representative traces for different traffic volumes and characteristics. The packet trace is replayed from an Android application sending uplink UPD packets to a server that replies with the downlink traffic. The traces are around 10 minutes long. Fig. 4.5 shows packet size, inter-packet interval (IPI) and the traffic volume characteristics of the traces. The name of the traces correspond to their traffic volume (low, medium and high). The traces have a majority of small packets and the transmissions range from regular (high) to sporadic (low) as the IPI shows in Fig. 4.5.

The traces described in Fig. 4.5 are used varying the maximum waiting time for a packet transmission ($T_w$) of the algorithm (30, 90 and 180 s). For every trace, we replay the trace without KLS as baseline. Then, the KLS is loaded and the traces are replayed for the different $T_w$. We measure the
4.4. EVALUATION METHODOLOGY

Figure 4.5: Characteristics of the emulated traces.

energy savings and CPU usage of the KLS as described below.

**Networking energy consumption:** The transmission energy consumption is conveniently calculated using EnergyBox. The 3G network settings we use correspond to the operator TeliaSonera in Sweden measured in our local area. The inactivity timers are $T_1 = 4.1 \, \text{s}$, $T_2 = 5.6 \, \text{s}$. The RLC buffer thresholds correspond to: $B_{u1} = 1000$ and $B_{u2} = 294$ bytes for uplink, and $B_{d1} = 515$ and $B_{d2} = 515$ bytes for downlink. The time to perform the different state transitions are set to 1.7 s, 0.65 s and 0.435 s for PCH-DCH, FACH-DCH and PCH-FACH respectively. We set the following power values for the different 3G states based on measurements [94]: DCH = 600 mW, FACH = 400 mW and PCH = 0 W. By setting PCH = 0 W, we aim to quantify only the energy spent in data transmission and energy tails.

**CPU energy consumption:** The CPU usage of the KLS itself is retrieved using the `top` utility. We perform physical measurements in the HTC Sensation to characterise the power usage for different CPU loads in a very simplistic manner. The measurements are performed by replacing the battery of the smartphone by a low-side sensing circuit (described in Appendix B). The frequency of the CPU is set to the maximum (1.2 GHz) in order to measure the highest power level, i.e., worst case. This also simplifies the quantification of the CPU power usage by avoiding the power variations due to dynamic voltage and frequency scaling. A CPU stress application is developed to increase the CPU load step by step up to 100% (7 steps). The CPU power is isolated from other factors such as screen, network interfaces or other applications by switching the components off and killing the processes that can interfere in the measurement. By interpolating the (CPU load, power) points we obtain a function that is employed to obtain the power consumption out of the CPU load data. The model used is presented in Figure 4.6

**KLS settings:** The settings for the KLS are set for the TeliaSonera network described above. In particular, we set $B_1 = 1000$ bytes, $B_2 = 294$...
bytes and $T_e = 300$ ms. $T_w$ is varied for the different tests (30, 90 and 180 seconds). The inactivity timers are used to simulate the state of the UE.

### 4.4.3 Measurement-based methodology

We evaluate the networking energy savings comparing them to TailEnder [18] as a baseline, the state-of-the-art algorithm to schedule background traffic. Directly measuring the transmission energy on the broadband module provides a fine-grained evaluation environment. This section provides a brief description of TailEnder, the data generation approach, the test framework used for the evaluation and a brief description of the algorithm implementations.

**TailEnder:** The algorithm schedules applications’ data transmissions based on user-specified deadlines. Each transmission is deferred until its deadline unless it arrives within $\rho \cdot T$, a time fraction of the tail-time (i.e., inactivity timer) of the most recent deadline. The algorithm is parametrised by the constant $\rho \in [0, 1]$ is a constant, which is set to 0.62 in their implementation. We use the same settings.

**Data generation:** We detail our trace gathering method. We used `tcpdump` in a Samsung Galaxy SII running Android 2.3 connected to the TeliaSonera operator in Sweden. In order to only allow the traffic from the desired applications we used a `iptables` based firewall named DroidWall\(^1\). We captured all the 3G communication for 40 minutes.

The applications running in the smartphone were Skype version 2.5.0.108 and Facebook 1.7.2. All the traces were gathered with the screen switched off and the application running in background. No user interaction was performed. The gathered trace has many small packets.

**TestFramework:** We developed a C++ TestFramework which runs in the test computer of our physical setup in order to run any packet-driven algorithm. The application allows us to replay any captured data traffic

\(^1\)http://code.google.com/p/droidwall/
trace and schedule the packet sendings to our 3G modem. The overview of the application flow is shown in Fig. 4.7 and is explained below.

The pcap (packet capture) files consist of previously gathered data traffic packets. The input parser uses \textit{TShark}\(^2\) commands to convert the gathered pcap traces into sequence of packet sendings with the following format: \{\textit{timestamp, packetsize, packetnumber, ip source, ip destination, protocol}\}.

The user simulation thread replays the original trace providing the input packets to the interchangeable algorithm in turn. We implemented a simple packet forwarder (simulating FIFO), our CLBB algorithm and TailEnder as the algorithms under test.

Finally, the network abstraction layer is a wrapper of network functionalities allowing the algorithms to send and receive packets to the server using simple functions.

**TailEnder and CLBB in the TestFramework:** Our algorithm is implemented in the TestFramework using two different threads (producer-consumer). The queues are implemented using C++ STL multimaps. Our TailEnder implementation is divided in two threads. The packet driven thread sends or queues a packet based on the following: if the UE is still within a time fraction of the tail time from the last sending it directly sends the new packet, otherwise the packet is queued. The other thread checks the shortest deadline of the queue and schedules transmissions when it is about to expire. Our implementation keeps track of the fraction of tail time by calculating the relative elapsed time from the last transmission.

### 4.5 Energy simulation results

The goal of this section is to calculate the total energy savings considering the networking energy savings and the cost of running the algorithm.

\(^2\)Terminal based Wireshark.

http://www.wireshark.org/docs/man-pages/tshark.html
4.5.1 Network transmissions energy savings

In this section we validate the energy savings of the KLS in the HTC Sensa-
tion for the 3 representative background traffic traces (High, Medium and Low) described in section 4.5. We present the energy savings of the UE with the embedded implementation of the CLBB algorithm. The replay of the captured trace without employing KLS is referred to original traffic and used as the baseline for calculating the percentage savings on the Y-axis in Figure 4.8.

Fig. 4.8 shows that the energy savings depend on the traffic volume: when the traffic volume is higher, the energy savings tend to be higher. The original traces consume 225, 161 and 105 Joules for the high, medium and low traffic volume traces respectively. Moreover, as expected, increasing $T_w$ leads to higher energy savings. For the low traffic volume trace, the energy savings range from 8% ($T_w = 30$ s) to 23% ($T_w = 180$ s). However, the energy savings are much greater (35% when $T_w = 30$ s) with more frequent background traffic. The maximum registered is 60%.

Fig. 4.9 shows the percentage of time spent by the UE in the different 3G states over the duration of the original traces compared to the case where the traffic is scheduled by the KLS. The KLS extends the time spent in PCH state which leads to energy savings. Intuitively, a greater $T_w$ makes the UE to be in PCH for longer period. In general, the amount of time in FACH is reduced in all the cases. For the original traces, it can be noticed that the UE spends more time in FACH with higher amount of traffic. Regarding DCH, the UE performs occasional burst transmissions with the High and Medium traffic profile, leading the UE to DCH for a short time period. However, with Low traffic profile, the UE rarely transitions to DCH.

To summarise, we validate that the KLS implementation of the CLBB algorithm provides high energy savings to the UE.
4.5. ENERGY SIMULATION RESULTS

Figure 4.9: Percentage of time spent by the UE in the different 3G states over the total time of the original traces (y axis) scheduled with different $T_w$ (seconds) by the KLS (x axis).

4.5.2 CPU energy footprint

In this section we show the CPU energy cost of running the KLS. The implementation performs two main operations, intercepting and queueing the packets, and sending them (dequeuing and re-injecting them in the network stack). The rest of the time the threads of the KLS are sleeping and therefore do not waste CPU time and energy.

In order to quantify the energy footprint when queueing packets, we queue several UDP packets from an application and observe the CPU load created by the KLS. The CPU usage of the KLS is logged and the energy footprint is obtained from the CPU load employing our CPU power characterisation explained in section 4.5. Our measured CPU load is below 0.01% which shows that the footprint due to the queuing operation is negligible in terms of CPU and therefore energy consumption.

Figure 4.10: Example of CPU energy consumption by running the KLS (left) and CPU energy footprint for the different traces (right).
Next, the CPU energy footprint due to sending the packets is measured for the high, medium and low traffic volume traces. Fig. 4.10 (left) shows an example of the energy consumption when running the KLS for $T_w = 30$ s for the high traffic volume trace. The KLS only creates CPU load spikes that are lower than 5%, which consume less than 0.25 W as it is shown in Fig. 4.10. These power spikes do not consume much energy.

Fig. 4.10 (right) shows the additional energy consumption of the CPU for the traces varying $T_w$. The maximum energy cost is 5 Joules, which is only 2% of the energy consumed for sending the original high traffic volume trace (225 Joules as shown in section 4.5.1). The energy footprint is higher for traces with higher traffic volume since the KLS has to process more packets. Except for the case of low traffic volume, a greater $T_w$ leads to lower energy consumption.

To summarise, we see that the maximum energy footprint of the KLS depends on the amount of traffic and $T_w$, but it is low compared to the energy used to send the original traffic due to energy-efficient kernel implementation of the algorithm.

### 4.5.3 Total energy savings

This section presents the total energy savings, i.e., the network transmissions savings minus the CPU energy cost of the KLS, i.e., the overhead. Since the energy cost of the KLS is small, the energy savings still remain high.

![Network transmissions energy savings, KLS energy overhead and total energy savings as a percentage of no-KLS base energy consumption.](image)

Figure 4.11: Network transmissions energy savings, KLS energy overhead and total energy savings as a percentage of no-KLS base energy consumption.

Fig. 4.11 shows the networking savings of the KLS, the energy cost of the KLS and the total energy savings as a percentage of the total energy consumption of the original traces (225, 161 and 105 Joules for High, Medium and Low traces respectively). When the traffic is low or moderate, the overhead is low and the total energy savings remain close to the networking energy savings. A somewhat higher energy overhead is observed for
the High data profile, but still the total energy saving is significant. In this case, the highest saving is achieved with the highest $T_w$, leading to a 58% net saving.

4.6 Measurement results

We now focus the study in thoroughly comparing the networking energy savings against the state-of-the-art algorithm using the broadband module for two different networks.

In order to calculate the energy savings, we measure the energy consumption baseline by transmitting the original trace gathered from the Android smartphone using the TestFramework. Then, for each of the points in Fig. 4.12 and 4.13 we ran our TestFramework with the original trace as input and measured the energy consumption using TailEnder and our algorithm separately. To compare the algorithms, $T_w$ is set to be the user specified deadline of TailEnder and it is equal for all the packets. The energy savings are computed with respect to the baseline varying the maximum waiting time $T_w$ for the packets.

Two different cellular operators are employed for the evaluation: Telia-Sonera and Hi3g Access AB under the name 3 (operator 1 and 2 respectively). The network parameters of the cross-layer burst buffering algorithm for operator 1 are $T_1 = 5$ seconds, $B_1 = 1000$ bytes, $B_2 = 294$ bytes, $T_e = 300$ milliseconds. Note that the value of $T_1$ is conservative, being longer than the actual inactivity timer.

For operator 2, we set the following parameters: $T_1 = 8$ seconds, $B_1 = 90$ bytes, $B_2 = 90$ bytes and $T_e = 600$ milliseconds. The $P_1$ and $P_2$ are set to 1.2 and 0.6 W corresponding to the measured power values.

![Figure 4.12: Energy savings of cross-layer scheduling and TailEnder for operator 1.](image)

Fig. 4.12 shows that the energy savings of the cross-layer scheduler are
higher than the TailEnder. The energy consumption baseline was 777 Joules of the original trace. The minimum energy savings in our measurements are 27% for this particular trace. It is worth mentioning that for the shortest $T_w$ our method achieves higher networking energy savings than larger $T_w$ for TailEnder.

The higher energy savings for the short $T_w$ of our method in comparison to TailEnder are achieved based on the fact that transmitting small packets in FACH is more energy efficient than triggering DCH state transitions. Our method takes advantage of the small packets avoiding the unnecessary state transitions to DCH.

For TailEnder, the larger the $T_w$ value the higher energy savings can be achieved. Since the traffic aggregation is performed with more data packets, the UE stays longer in the PCH state. The same issue arises for the cross-layer burst buffering when increasing the $T_w$. However, the energy savings of our method decreases for applications with longer $T_w$. The cause of this fact is that the probability of having bigger packets in the queues increases when increasing the $T_w$ and therefore the possibilities of performing sendings in FACH are lower.

The gap in Fig. 4.12 can further increase by network settings that have greater $B_2$ or shorter $T_e$, i.e., higher available data rate in FACH. Further, data sets with smaller packets would be more beneficial to our system; as it would be the case with a longer $T_1$.

Operator 2 employs smaller $B_2$ and allows lower data rates in FACH. Fig. 4.13 shows the networking energy savings results for operator 2 (baseline 1120 Joules). Even though operator 2 has a very small $B_2$ and larger $T_e$, the energy savings of CLBB are still higher since it still performs transmissions in FACH. The gap between both algorithms is decreased since the value of $B_2$ allows fewer transmissions in FACH and the data rate in FACH is lower.

The energy savings provided by the algorithm compensate for the cost of the algorithms to infer the network parameters. This cost of was measured to be 95 Joules in section 4.2.3, which is already compensated in the 40 minutes trace of the test.

Finally, Fig. 4.14 shows a fragment of one of the runs of CLBB using the network of operator 1 for $T_w = 90$ seconds. The example shows that by aggregating transmissions CLBB keeps the interface in idle until a packet needs to be sent. Instead of transmitting the aggregated packets in a burst that would lead to DCH, the example illustrates how the transmissions are performed in the FACH state reducing the energy consumption due to the wireless interface.

### 4.7 Exploring live traffic

Next, as an exploratory phase, we evaluate the KLS implementation with applications that create live traffic. In general, applications do not distin-
4.7. EXPLORING LIVE TRAFFIC

Figure 4.13: Energy savings of cross-layer scheduling and TailEnder for operator 2.

Figure 4.14: Energy consumption examples of background traffic and energy-efficient shaped background traffic.

guish their background traffic from the rest of the traffic, neither do they specify which traffic could potentially be shaped or delayed. This results in the KLS (or a similar approach) scheduling all outgoing traffic. Nevertheless, we test the KLS in the current settings to study the potential implications of performing traffic shaping during live operation of the same applications.

When capturing different traces for evaluating the KLS with live traffic, it is complicated to compare their respective energy consumption since applications have different traffic pattern at different times. To overcome this limitation, we capture traces with and without the KLS to compare the standard traffic behaviour against the behaviour with the KLS. Facebook and Skype are used as test applications with and without the KLS.
(T_w=30s). The screen of the smartphone is switched off and we do not actively use the device.

**Skype:** the UE experiences a significant increase in traffic when running Skype with the KLS (51 packets without the KLS against 1145 packets with the KLS), leading to higher energy consumption. We identify several factors that explain the increase of the traffic, which are dependent on the application’s live operation or behaviour.

The Skype client periodically builds a host cache (IP address and port) of Skype servers (supernodes) using the DNS protocol. The Skype client sends a UDP packet of 18 bytes to the port 33033 of a supernode on its host cache. When there is no response for 5 seconds, the client attempts to establish a TCP connection. In case of failure, this is repeated every 6 seconds. When the KLS is running, this operation increases greatly the amount of traffic. Moreover, the delay introduced by the KLS also affects the Skype’s TCP operation. In particular, the TCP retransmissions, out of order packets and duplicate ACKs are 8 fold higher when using the KLS.

**Facebook:** we observe a favourable behaviour for the KLS when running Facebook. The packets are scheduled according to the algorithm in an energy efficient manner. Even though Facebook also uses TCP, no extra retransmissions, out of order packets or duplicated ACKs are observed and the amount of traffic is similar with and without the KLS.

To sum up, our observations reveal that shaping applications’ outgoing traffic independently from their intention/behaviour can result in a counter-productive operation of some applications (e.g., Skype), whereas for others it can work out of the box (e.g., Facebook). In any case, if we do not make traffic distinction we cannot exploit the energy saving potential for all applications. Background traffic differentiation becomes an important piece of the energy puzzle in order to fully achieve energy savings.

### 4.8 Discussion

Achieving energy proportionality in systems that were designed for peak performance is a complex task. The focus of this chapter was to develop a generic energy saving solution that can provide energy proportionality during low utilisation periods given flows with no QoS requirements. These transmissions allow to alter their data pattern to create energy-efficient traffic.

This chapter has shown that the EnergyBox is indeed a valuable tool for evaluating energy savings solutions. EnergyBox has efficiently quantified the networking energy savings of the proposed solution.

The knowledge of the hidden energy footprint characteristics of the radio communication layer becomes valuable when developing energy efficient solutions to optimise the usage of the battery resource. In this chapter, our scheduler uses a more detailed radio communication layer knowledge to improve the energy savings results of the state-of-the-art. Moreover, we study
the cost of running the solution itself and show that applying a mechanism that employs a small amount of energy to save energy pays off.

The maximum time to wait for a transmission is employed by the algorithm to delay transmissions. The parameter is provided by applications or the operating system. While our evaluation used several values ranging from seconds to minutes, this can be adjustable by the user as a trade-off between energy and delivery delay. The maximum time to wait can be increased using inactivity time as an estimator, knowing that when the user has been idle for some time it is more likely that the user will continue inactive [35].

In this work we do not consider fragmentation nor the use of the radio link quality estimator like RSS to adapt our data transmissions. Packet fragmentation would allow to reduce the packet sizes to be transferred in lower consuming states. Considering link quality would further increase the energy savings by transferring during good link periods when the transmissions cost less energy.

Nevertheless, as a part of our exploratory phase with real live traffic, we discover that the mix of data and background traffic cancels the energy saving operation of our kernel module (intended only for background traffic). However, there is evidence that for some low-intensity applications this would work out of the box.

The question of packet differentiation for energy saving is an open field for further exploration. Applications (and therefore developers) could be in charge of differentiating the background traffic from the rest. However, great part of the background traffic is created by applications due to not being aware of the transmission energy characteristics. Traffic differentiation without application cooperation is an interesting approach to investigate, which could be performed at different layers to maximally exploit the energy saving potential.

For traffic types other than background, energy-efficient solutions need to consider the requirements of the traffic or applications, which implies that the traffic data pattern cannot be completely modified.
Chapter 5

Energy-Efficient Location Sharing

In earlier chapters we saw that the transmission energy consumption at the user end is highly affected by the data pattern. The viability of shaping the data traffic without any strict requirement creating more energy-efficient traffic patterns was shown in the previous chapter. However, the unpredictability of users and their interaction with applications as well as the application requirements constrain the room for generic energy saving solutions based on traffic class.

Nevertheless, domain knowledge of certain application categories and the generated data pattern enables the creation of tailor-made energy-efficient solutions. The energy consumption of specific applications can be minimised at the design stage by performing the correct choices that impact the data pattern (e.g., protocols or payload size). When analysing the energy consumption of the different design choices, a tool like the EnergyBox becomes valuable and convenient to adopt the energy-efficient options. In particular, we show that modifying the application protocol and the data format, i.e., changing the sending frequency as well as the size, the transmission energy can be drastically reduced.

The use of a mobile device’s battery for frequent transmissions of position data in a location sharing application can be more expensive than the location retrieval itself [55]. This is in part due to energy-agnostic application development and in part dependent on choice of protocols.

The scenario considered in this chapter is the following: location-based services are applications integrating geographic location with services, such as navigation systems, sports trackers or company and business information delivery. There are countless applications and services that support the user for sharing their location data with other users (e.g., Foursquare, Google Latitude or Gowalla), also referred to location sharing services. The location data is often shared over wireless networks that provide ubiquitous
connectivity, such as cellular networks. Fig. 5.1 shows an overview of data flow in a location sharing service.

![Diagram of data flow in location sharing service](image)

Figure 5.1: Overview of the data flow in a location sharing service based on cellular (3G) communication.

In this chapter we isolate the energy overhead for location sharing which arises due to the cellular network data transmission regime. We show that using EnergyBox we can efficiently compare the energy consumption of application design choices in terms of application layer protocols or data formats. In particular, we compare the commonly used Hypertext Transfer Protocol (HTTP) and the Message Queueing Telemetry Transport (MQTT) protocol as an alternative. MQTT follows the publish-subscribe paradigm whereas HTTP is a client-server protocol. The implementation of a location sharing application developed for the Master thesis of Mihails Prihodko [88] is used to experimentally collected real data and perform the comparative study. This chapter is mostly based on the material from that thesis which was performed under author’s supervision.

The chapter is organised as follows: section 5.1 provides a description of the MQTT protocol. The location sharing application developed for experimentally comparing the protocols is presented in section 5.2. The evaluation methodology is described in section 5.3, and the results are presented in sections 5.4 to 5.6. The results are discussed in section 5.7.

### 5.1 MQTT protocol background

The MQTT protocol is an extremely lightweight and simple publish/subscribe messaging protocol designed for machine-to-machine (M2M) or “Internet of things” contexts, e.g., constrained devices used in telemetry applications. The aim of MQTT is to minimise network bandwidth utilisation (the fixed-length header is 2 bytes) and device resource requirements while attempting
to ensure some level of reliability. The version 3.1 of MQTT specification is the latest and openly published specification with a royalty-free license [2].

The publish/subscribe communication paradigm is based on subscriber entities that express their interest in certain information or events, which are provided by publisher entities. Published messages are categorised into classes (topics). A subscription is defined as the registration of interest by the subscriber in one of these classes. Subscribers are instantaneously notified about every new event generated by the publisher that matches their interest, and they only receive the messages related to their subscription, i.e., a subset of the total number of messages published. According to the MQTT specification [2], a client is a piece of software that can behave as a publisher and/or a subscriber. The clients communicate via a server or broker, i.e., the piece of software that matches publisher messages to subscribers. The design of MQTT allows a small code footprint implementation of the client side, moving the complexities to the broker side.

The broker decouples publishers and subscribers along space and time: the sender does not have to know the number, location or identifiers of the receivers, and the participants do not have to interact at the same time. Events can be published even before subscribers are connected, and the subscribers are asynchronously notified even when publishers are disconnected.

MQTT employs topics encoded as character strings to filter the messages. Fig. 5.2 shows an example of MQTT operation for two topics, where the broker only sends the messages to the subscribers with a subscription to a particular topic (e.g., subscriber 1 only receives messages about topic 1).

MQTT assumes that the underlying network layer provides a point-to-point data transport service with in-order delivery such as TCP/IP. However, in order to increase reliability, MQTT also defines 3 levels of Quality of Service (QoS) that implements additional mechanisms above the transport level to guarantee message delivery (e.g., acknowledged delivery or no message duplicates).
A MQTT client (publisher or subscriber) needs to establish a connection with the broker before any data exchange is performed using a *CONNECT* message. This message provides a keep alive timer used by the broker to detect connection failures without waiting for lower layer protocol timeouts (e.g., the long TCP/IP timeout). The client sends a *PINGREQ* message within the keep alive when there is no data transmission. The client is disconnected when the broker does not receive any message within 1.5 times the keep alive period. The value employed for the keep alive timer depends on the application, which can range from few to several minutes.

## 5.2 Location sharing application

Since current location sharing applications are closed systems (products), the behaviour of the real systems cannot be compared against MQTT in the same settings. This section describes the design and implementation of a Location Sharing Application (LSA) developed to perform a comparative energy consumption study of the HTTP and MQTT protocols. The LSA is used as a tool to experimentally evaluate HTTP and MQTT providing means to exercise control over the experiments.

The LSA is designed to share location information among different users connected to the Internet via an UMTS network. The users can use it in check-in and tracking mode. In check-in mode, the location is updated based on user demands (typically sparse intervals), whereas tracking mode allows frequent location updates. The LSA consists of a client side and a server side that can communicate using two techniques. TCP is used as transport layer protocol for both techniques.

- **HTTP technique**: Since HTTP functions as a request-response protocol using the client-server paradigm, the client polls the server in order to retrieve other clients’ location information updates. A text-encoded location object is sent between the client and the server. This technique mimics HTTP-based existing location sharing applications and serves as a baseline for comparing the energy consumption. The operation of the HTTP technique is similar to a pull notification mechanism\(^1\).

- **MQTT technique**: The publish/subscribe paradigm implemented by MQTT is used as an alternative protocol for LSA. The location data is a text-encoded object or a binary object (depending on settings) sent between the publisher/subscriber and the broker. The operation of the MQTT technique is similar to a push notification mechanism.

\(^1\)While push/pull notifications can be implemented in different ways using the same protocols (e.g. HTTP server push or private push services), we choose to use HTTP and MQTT in their standard form for simplicity since their standard usage is provided by the standard APIs.
Architecture: Fig. 5.3 shows the general architecture of the LSA. The client and the server are modular Java applications with interchangeable elements that allow the usage of both techniques. The client was developed for the Android platform using the API provided by the Apache HTTP client and the IA92 MQTT library.

For the server side, we developed a Jetty-based\(^2\) HTTP server implementing a custom request handler for the specific client HTTP requests. For the MQTT technique, the same server uses the Really Small Message Broker (RSMB)\(^3\). The MQTT handler and the HTTP handler are the interface between the RSMB and the user simulator. The MQTT handler runs any number of MQTT clients created by the user simulator as described later.

\(^2\)http://www.eclipse.org/jetty/

\(^3\)http://mqtt.org/wiki/doku.php/really_small_message_broker
5.2. LOCATION SHARING APPLICATION

**Client-server side interaction:** The client and server side interaction differs in the two techniques. In the client side implemented in the mobile device, the GPS location provider creates an event roughly once every second while the device is moving. The sharing period can be chosen independently from the location provider update period, that can depend on current application requirements such as accuracy or energy efficiency [55]. The location sharing update period is denoted as $T$. Fig. 5.4 shows how a single location sharing update is performed for both techniques. Whenever the client updates its location using HTTP, it also retrieves other users’ location updates. The location update of the publisher is instantly forwarded by the broker to the subscribers when using MQTT.

**User simulator:** It simulates the location sharing activity of any number of users in our experimental evaluation. The simulator is designed to mimic the behaviour of users constantly moving, i.e., updating their position once every period $T$. We define asynchronous (async) and synchronous (sync) location sharing behaviours described below.

Since the HTTP technique polls the server to retrieve users’ location updates, the locations are sent in bunches by performing synchronous location sharing. The user simulation creates a new location for every user once every $T$. The default behaviour of the MQTT broker is asynchronous since the broker forwards the published updates to the subscribers as soon as they reach the broker. The simulated publishers send their location update once every $T$ using a random offset to mimic real user location sharing behaviour. Moreover, in order to study the synchronous and asynchronous location update behaviour of MQTT and its impact on the data pattern and energy consumption, we design the server in a way that it can perform asynchronous location sharing. The server can hold on to the updates and forward them every specified period $T$ in a single burst.

**Location data and encoding:** We selected a subset of the values provided by the Android location object based on the study of the API of a popular application\(^4\), and included time, latitude, longitude, speed, accuracy, altitude and bearing in our location object. Since the data encoding influences the data pattern in terms of packet size, we also consider different data encoding formats in our study. JavaScript Object Notation (JSON) is a text-based human-readable format used as de facto standard for mobile-based applications. In the search of more compact formats, we also encoded the location data in Base64 Content-Transfer-Encoding format (base64). Base64 is a format designed to represent arbitrary sequences of octets in a text-based non-human readable form. The sizes of the different types of formats are 154 and 61 bytes for JSON and base64 respectively.

\(^4\)Google Latitude API
5.3 Evaluation methodology

This section describes our methodology using EnergyBox as an energy estimation tool. The LSA architecture was used to analyse the energy consumption and the data pattern, varying the number of users in the system (different network load), location sharing update interval (T), and data encoding. Analysis of the energy consumption, assisted by EnergyBox, allows exploration of the design space for reducing the energy footprint.

The overview of the evaluation is as follows: Section 5.4 reports the experimental energy consumption study of MQTT and HTTP, and based on the results section 5.5 studies the appropriate protocol for check-ins employing a user usage knowledge from a public dataset of location sharing service users. Finally, section 5.6 studies whether HTTP can benefit from using a compact data encoding format that reduces the packet sizes.

We perform our experimental study by using an Android smartphone (LG P990) with a GPS receiver as a UE connected to a 3G network of the TeliaSonera operator in Sweden. The experiments are emulating real user location traces, performed walking outdoors while carrying the smartphone with GPS signal over a predefined path at the university campus. The idle experiments were performed indoors where no GPS signal is available and the smartphone is stationary to avoid location updates. We simulate a number of users with whom the client shares location updates using the user simulator. The length of each experiment is 10 minutes. An iptables based firewall is employed to allow only the traffic created by the LSA. We capture the data traffic of the LSA using tcpdump in the smartphone, which is later used to quantify the energy consumption as explained below.

**LSA settings:** The location sharing update interval (T) is increased in multiples of 15 seconds (15, 30, 45 and 60 s) to study the impact of the update frequency on the data traffic and energy consumption. In order to study the impact of higher traffic load, the number of simulated users (active friends sharing their location data) in the system is set to 0, 3, 6 or 9. The settings for the data encoding formats are specified for each experiment.

**MQTT settings:** The QoS of MQTT is set to Level 0 relying only on the transport layer protocol (TCP) message delivery mechanisms as HTTP does. Since HTTP does not provide any connection failure detection mechanism, we set the MQTT keep alive timer to 30 minutes (higher than the experiment time) to compare both protocols with similar settings.

**Networking energy consumption:** The transmission energy consumption is calculated from the gathered traces using the EnergyBox. In order to calculate the energy consumption of the traces we collect during the experiments, we set the 3G network settings that correspond to the operator TeliaSonera measured in our local (experiment) area.

The inactivity timer settings correspond to \( T_1 = 4.1 \text{s}, T_2 = 5.6 \text{s} \). The RLC buffer thresholds correspond to: \( B^u_1 = 1000 \) and \( B^u_2 = 294 \) bytes for uplink, and \( B^d_1 = 515 \) and \( B^d_2 = 515 \) bytes for downlink. The time
to perform the different state transitions are set to 1.7 s, 0.65 s and 0.435 s for PCH-DCH, FACH-DCH and PCH-FACH respectively. The UE power values for the different RRC states are based on earlier measurements [94]: DCH = 800 mW, FACH = 460 mW. We set PCH = 0 W in order to quantify only the energy spent in location sharing data transmissions.

**Evaluation metrics:** We employ the percentage of time spent by the UE in different RRC states over each experiment time, and the total energy consumption (Joules) as energy consumption metrics. We also study the total amount of data sent over the experiment. All the figures are normalised through dividing the results by the greatest values resulting in the figure and providing the base values.

### 5.4 HTTP vs. MQTT

In this section we compare the HTTP and the MQTT protocols using the same data format (JSON), update period and number of simulated users. Since asynchronous location sharing is the default behaviour for MQTT, the simulated users update their location asynchronously. We differentiate the idle state (no location updates by any of the users in the system) from the active state.

**Idle state:** The UE may spend significant part of its time in idle state between aperiodic or infrequent location sharing updates (e.g., check-in mode or infrequent events). Fig. 5.5 shows the energy consumption and amount of data traffic generated by the HTTP and MQTT protocols in the idle state, where no location data is exchanged. The data traffic and energy consumption values are normalised to the greatest values, 18.59 kB and 143.37 Joules.

![Figure 5.5: Amount of data traffic and energy consumption for HTTP and MQTT in the idle state.](image-url)
The HTTP (pull) technique creates more traffic than the MQTT (push) technique. The HTTP client needs to poll the server in order to find out whether new location updates are available. In the client-server paradigm, the client needs to start the communication. The update period $T$ strongly influences the data traffic, which gradually decreases when increasing $T$. The MQTT technique only establishes a connection with the broker at the beginning of the experiment, and therefore the data traffic is constant for all $T$. The main benefit of the publish/subscribe paradigm employed by MQTT is that the broker can send a location sharing update whenever it happens without polling the server, irrespective of the update period.

The HTTP technique also consumes more energy than the MQTT technique. The HTTP client needs to poll the server even when no new location data is available. The longer the update period $T$ is, the smaller the energy consumption becomes. However, the delay between the users’ location updates and the reception of them becomes greater. Since the HTTP transmissions do not contain any location data, the data only triggers the FACH state and make the UE consume one energy tail. The energy consumption of MQTT is constant. The UE only switches to FACH for the connection establishment and then spends the rest of the experiment in PCH.

**Active state:** In the idle state experiments the energy was mostly consumed by polling or preparedness to send. We now study the behaviour of HTTP and MQTT when actual location data is transmitted. The goal of the experiments is to understand the operational needs of the application so that the most energy efficient protocol can be chosen according to the requirements. We vary the update period $T$ and the number of simulated users that the client shares location data with, i.e., the load in the system. The case of 0 simulated users represents when only the client updates its own location.

![Figure 5.6: Amount of data traffic for HTTP and MQTT in active state.](image)

Fig. 5.6 shows the amount of data transmitted by the UE normalised to the greatest value (125.24 kB). We split the results into two figures for read-
ability. Fig. 5.6 (left) shows that MQTT generates lower traffic comparing to HTTP when the number of users is low, independently of the update period. The small overhead of the MQTT protocol is the main differentiating factor for low number of users. A single location update performed by the client consists on average of 421 bytes and 1159 bytes, for using MQTT and HTTP respectively.

Fig. 5.6 (right) shows that the MQTT and HTTP techniques generate similar amount of traffic when the number of users is 6. Somewhat surprisingly, the MQTT technique creates more traffic for 9 users than HTTP. HTTP retrieves all the location updates in bundles every update period, whereas the MQTT publish/subscribe updates are transmitted whenever they happen. Sending the updates in small packets separately increases the protocol data overhead. For example, the many small packet transmissions of MQTT make the total TCP overhead increase by up to 23% (9 users and an update period of 15 seconds), whereas it is only 3% for the HTTP technique as a result of bundling the updates.

Fig. 5.7 shows the packet size and inter-packet interval (IPI) distributions for the case with 9 users and an update period of 15 seconds. Even with the same data encoding, the packet size for the MQTT technique is smaller as it is shown in empirical cumulative distribution function (CDF) of the packet size in Fig. 5.7 (left). Fig. 5.7 (right) shows that the IPI is typically shorter for HTTP. Most of the packet transmissions of the HTTP technique are bursty transmissions involving acknowledgements with short IPIs, whereas for MQTT the packet transmissions are more sparse. The other factors that impact the empirical CDF of HTTP are the PCH-DCH transition delay and the update interval T. Note that the MQTT technique is not affected by the PCH-DCH transition delay since the transmissions are mostly performed in FACH (the PCH-FACH transition delay is much shorter than the PCH-DCH transition delay).

![Empirical CDF of packet size and inter-packet interval of HTTP and MQTT (9 users and T=15s scenario).](image)

Figure 5.7: Empirical CDF of packet size and inter-packet interval of HTTP and MQTT (9 users and T=15s scenario).

The energy consumption for HTTP and MQTT is shown in Fig. 5.8.
normalised to the greatest value of MQTT with 9 users (285 Joules). For HTTP, we show the energy consumption when only the user uploads her location data (0 users) and the average when there are more users in the system. The energy consumption of HTTP is similar for 3, 6 and 9 users, and therefore we only show the average value to compare with MQTT (see “HTTP - with users” in Fig. 5.8). Since the amount of data transmitted every update period using HTTP is always greater than the PCH-DCH RLC buffer threshold, the UE is moved to DCH. Hence, the number of energy tails caused by the inactivity timers dominate the energy consumption. Therefore, the longer the update period the lower the energy consumption.

Figure 5.8: Normalised energy consumption for HTTP and MQTT in active state.

For 0 users (when only the client updates it own location regularly), the energy consumption of HTTP and MQTT are similar. Fig. 5.8 shows that MQTT is more energy-efficient for shorter update periods when the number of users is 3. When the update period is increased for 3 users, HTTP consumes less than MQTT since it updates the location of the users once every T. The HTTP protocol becomes more efficient when the number of users increases (6 and 9). The reason is that HTTP transmits all the data in a single burst (polling nature), at the cost of location sharing delay (every T).

Fig. 5.9 shows the fraction of time spent by the UE in the different RRC states over the total experiment time. Both techniques have a similar radio resource utilisation with 0 users. The UE performs transmissions mostly in FACH, leading to the FACH energy tail after every update. When the client employs HTTP with more users in the system, the transmissions are performed in DCH and the UE experiences the overhead of the DCH and
5.5. Protocol for check-in

We are interested in studying the time between updates (inter-update interval) for each user as an input to guide choosing between the two protocols. In this section we study the appropriate protocol for check-ins, i.e., the process whereby a user reports her presence or arrival at a location.

We employ the check-in dataset provided by Cheng et al. [27]. The

FACH energy tails. Instead, MQTT makes an efficient use of the FACH channel due to its low overhead while there are few users (3 user case). However, when the number of updates is increased, the FACH occupation is increased as well (6 and 9 users). Higher volume in MQTT location updates force the UE to move to DCH.

To sum up, the results show that the low protocol overhead and the publish/subscribe nature of MQTT makes MQTT more energy efficient in idle state and when the number of users is low. When the load is increased, the HTTP technique takes advantage of aggregating all the location updates in a single burst being more energy-efficient. However, this comes at the cost of delaying the reception of the updates at the client side. Given the application usage (e.g., number of expected users or expected update period) the protocol choice can greatly reduce the energy consumption.
dataset contains 22,387,930 check-ins from 224,804 location sharing service users gathered from the public Twitter feed between September 2010 and January 2011. The check-ins dataset contains check-in information in tuples of \{userID, TweetID, Latitude, Longitude, CreatedAt, Text, PlaceID\} and user information \{userID, StatusCount, FollowersCount, FriendsCount\}. We calculate the inter-update interval for each user using the above dataset.

Fig. 5.10 shows the empirical CDF of the inter-update interval of all the users in the dataset. Fig. 5.10 (left) shows a general view in hours and Fig. 5.10 (right) describes the same empirical CDF for shorter location update intervals in minutes. The inter-update interval is long for check-ins, which makes the idle time to dominate the active time. It also presents a long tail (not fully shown in Fig. 5.10) that describes the sparsity in the check-ins of many users. Fig. 5.10 (right) shows that inter-update intervals shorter than 1 minute represent 5.6% of the total number of check-ins, and a single user rarely has more than 1 short consecutive updates per 1 minute (93% of the users do not have any consecutive update shorter than 60 seconds).

In the previous section we saw that when the traffic load is small, MQTT was more energy efficient than HTTP and resulted in similar energy when the period increased. Therefore, given that the majority of check-ins show long inter-update intervals, MQTT appears to be the protocol of choice due to its instant location update delivery.

5.6 Data encoding impact on HTTP

Earlier, we stipulated that transmission energy of an application is greatly dependent on the data pattern. In this section we quantify the benefit of using a compact data encoding format (base64) over the standard verbose
format (JSON). The data encoding format impacts the size of the payload, and therefore the data pattern in terms of packet size. HTTP is selected as protocol for the comparison since it is more commonly used for transferring JSON data than MQTT (MQTT typically uses the binary format). The comparison is performed using the scenario with 3 users, emulating the case where the client is to be notified of three friends locations continuously.

![Figure 5.11: Normalised energy consumption and user equipment RRC states for different data encodings.](image)

Fig. 5.11 (left) shows the energy consumption normalised to the greatest value (283 Joules). The energy consumption of HTTP is significantly reduced when using a more compact format. The location sharing is performed in a single burst of packets in both cases. However, the reduced size of payload stops the UE from moving to DCH (Fig. 5.11 (right)), whereas the data transmissions of HTTP with JSON are performed in DCH (Fig. 5.11 (centre)). A single big packet is enough to trigger the FACH-DCH state transition based on the settings of the RLC thresholds, which leads to the DCH and FACH energy tails.

The results show that the energy consumption can be reduced by 16 to 50% when using a compact data encoding format at the cost of losing human readability.

### 5.7 Summary and discussion

In the search for tailor-made energy-efficient solutions for application data transmission, different design choices need to be systematically studied. Application protocol or data format choices impact the data pattern of the application, which in turn has a substantial impact on the energy consumption.

As an example to illustrate such choices, this chapter compared the MQTT and HTTP protocols as suitable location sharing protocol choices over 3G networks. The publish/subscribe nature of MQTT ensures instant
delivery of location updates, whereas an HTTP client needs to poll the server to retrieve updates creating a delay that is dependent on the update period. We show that MQTT creates less data traffic, is more energy-efficient in the idle state and when the number of users with whom the client shares its location is low. When the number of users (simultaneously sharing users) increases, HTTP becomes the preferred option since the location updates are aggregated when the client polls the server.

Given the above results and the knowledge of the usage type of a location application, one can select the suitable protocol to meet the application requirements. Another functionality that impacts data pattern is the check-in mode. Our analysis shows that with check-in mode MQTT appears as the protocol of choice. Other use cases (e.g., continuous location tracking) expose an energy-delay tradeoff. A further aspect that impacts the data pattern in terms of packet size is the data encoding format. Our results show that the energy consumption when using HTTP is reduced if the location data encoding format is compact.

Based on the presented results, the current work can be extended by developing an adaptive protocol switch (MQTT to HTTP) or switching the protocol mode (aggregating location updates for MQTT when the load is high) based on the number of users in the system to reduce their energy consumption. The energy overhead of switching the protocol mode, the location sharing delay and the energy savings would be interesting to study in a real application.

Finally, this chapter shows that EnergyBox or similar energy simulation approaches can become valuable instruments in order to choose the most energy-efficient options at design stage. However, the question of how to select among the wide range of real configurations to provide a representative and generic energy design or test environment is still an open question.
Chapter 6

Energy-Efficient Instant Messaging

When the transmission pattern is mostly determined by user interactions, the complexity of designing energy-efficient transmissions increases. This is very obvious in the case of Instant Messaging (IM) applications, where interactive traffic is generated by a sequence of messages exchanged by users in a conversation. Traffic generated by IM applications is usually elastic, since no QoS support is assumed for delivering the text messages. IM messages are an example of a type of traffic with low bandwidth requirement, but they lead to high energy consumption due to the radio resource allocation of cellular networks. The exchange of a couple of text messages can consume as much as sending an image.

IM applications have emerged as the substitute for Short Message Service (SMS) and have gained a wide popularity. These applications usually offer the possibility of sending text messages (1-to-1 or to a group) as well as other multimedia messages (e.g., images, audio or video) for free. Recently, IM on chat applications has overtaken the traditional SMS text messages in total number of messages sent per year\(^1\). IM applications provide more functionalities than regular SMS, such as online presence awareness, typing notification or status updates. Application developers integrate these features without knowing the a posteriori potential impact on energy consumption. Recall from section 2.1 the great diversity of energy consumption of the different IM applications tested when sending the same conversation. We argue that the energy consumption of the added features need to be studied at the design stage in order to be aware of the correct operation choices (e.g., parameters) and the potential impact on the energy consumption, and therefore QoE. EnergyBox or a similar tool is suitable for performing such studies.

\(^1\)http://www.bbc.co.uk/news/business-22334338
Knowledge of the user interaction pattern and its impact on the data pattern can enable the design of tailor-made energy-efficient solutions. Usage patterns can reveal inefficient ways of performing the transmissions since neither the users nor the applications are aware of the energy footprint characteristics.

In this chapter the suitability of EnergyBox to aid in energy consumption studies is demonstrated in two ways: (1) isolating the cost of a feature that most IM applications implement, and (2) evaluating a bundling technique to reduce the energy consumption with some knowledge of the of the user interaction pattern. While presence updates have been already studied in desktop-oriented machines [118] and in the mobile context [29, 71], the cost of the typing notification feature used by most chat applications has been neglected. EnergyBox estimations are used to show that this functionality does not come cheap when using the third generation cellular communication network. Furthermore, given the usage data collected from several users from an IM application, we observe the possibility of reducing the energy consumption by aggregating messages.

The chapter is organised as follows: section 6.1 describes an IM dataset of real user messages used to get an understanding of the usage of IM applications, and later used as evaluation traces. The algorithm used to test the typing notify feature is explained in section 6.2. Section 6.3 describes the proposed message bundling technique. The evaluation methodology using EnergyBox is presented in section 6.4 describing also the IM application developed for the tests. The energy cost of the typing notification functionality is studied in section 6.5. Section 6.6 evaluates the energy savings of aggregating messages and analyses the introduced delay. Finally, the results are summarised and discussed in section 6.7.

The development of the IM application and the experimental tests reported in this chapter were performed by Simon Andersson as a Bachelor thesis [11] under the supervision of the author.

### 6.1 Instant messaging dataset

A set of messages was collected from one of the most widely used IM applications (WhatsApp). A Java parser was developed to obtain the conversations from the WhatsApp logs. A conversation in the dataset is a sequence of messages with a timestamp (UNIX time), the message length in characters, and a unique number assigned to the user. The user name and the actual content of the message are obfuscated for privacy reasons. The focus of this work is on 1-to-1 text conversations, and therefore we do not consider group chats or multimedia messages which are left for extensions of this work.

The dataset currently contains 3726 messages spread in 15 conversations from 17 users during a period between 7th of January 2013 and 15th of May 2013. Note that different conversations have different starting time. The timestamp and the message size are the interesting values from the
transmission pattern perspective. We agree on collecting the usage data with the users after this was created (from past logs), thus the data reflects the normal behaviour of the users. The users employ WhatsApp as their primary IM application.

We have observed that users generally write more than one message in a row before getting any answer. 57% of the messages in the dataset show this characteristic. From the energy perspective, consecutive messages are wasteful since they restart the 3G inactivity timers leading to higher energy consumption. If the elapsed time is greater than the inactivity timer, the UE would consume another energy tail. The data suggests that consecutive messages can potentially be sent together increasing the idle time. Therefore, we also investigate the distribution of the subset of messages which includes consecutive messages, and refer to it as consecutive messages.

![Empirical CDF of inter-message interval and message size](image)

*Figure 6.1: Empirical CDF of inter-message interval and message size of the dataset.*

The empirical CDF of inter-message interval (IMI) and message size of the dataset is shown in Fig. 6.1, left and right respectively. Regarding all the messages in the dataset, we observe a long tail in terms of IMI. 25% of the messages have an IMI longer than 5 minutes. The mean IMI for the dataset is 21 seconds. According to the data, short IMI are more typical. The consecutive messages present shorter IMIs than the total set of messages, with a mean of 16 seconds. 70% of the consecutive messages have shorter IMI than 1 minute. The short IMI of consecutive messages suggest that these could be aggregated together.

Two types of *conversation fragments* (i.e., a chunk of the whole conversation) can be identified over time as well. Based on the number of messages sent and their closeness in time, we can define two different type of conversations over a period of time: *sparse* and *dense* conversation periods. Sparse conversation periods typically have their messages more separated in time (i.e., higher IMI), while the dense ones are chat intensive periods with shorter inter-message intervals. This knowledge is used later on to select the
evaluation traces.

Regarding message size, we observe that short messages are predominant. The average message size is 32.37 characters. The messages shorter than 40 characters are 73% of all the messages. Small messages lead to small packets, which makes very inefficient to send each message in a separate packet. Given that the transmission is performed over TCP/IP, the overhead of the headers (40 bytes) has usually the same size as the payload.

In order to understand which message sizes produce more traffic, we multiply the message size by the number of messages. Fig. 6.2 shows the distribution of traffic over the message size. Most of the traffic is generated by the small packets in the case of IM.

![Figure 6.2: IM traffic of the dataset.](image)

To sum up, the dataset provides valuable insight for studying efficiency of transmissions. Since most of the traffic is created by small packets and the presence of consecutive messages is significant (57% of all messages), aggregating consecutive messages can alleviate the energy consumption of IM applications. An algorithm to aggregate messages is presented in section 6.3.

### 6.2 Typing notification

The typing notification is a common feature implemented by most IM applications. The mechanism notifies the user when the other user in the conversation is typing, creating a notion of presence and interactivity. Users can use this information to decide whether they should remain in the conversation or not.

In order to study the cost of this mechanism, we develop an algorithm which emulates its operation in a simple IM application. The notify updates should be triggered when the user types a character in the text input area on the graphical user interface. However, it is up to the implementation to decide the frequency of these events resulting in a network packet being sent to the other user as a notification.

Algorithm 5 employs a countdown timer named NotifyTimer to restrict the rate of notify messages and thus prevent a packet flood. The NotifyTimer
is statically configured with a NotifyTime value in seconds. When the user is typing, a text input area changed event is triggered. If the NotifyTimer is not running (i.e., NotifyTimer = 0), the algorithm will send a notification packet to the network and start the NotifyTimer right after. The next time the user types, the NotifyTimer will avoid a new packet being sent to the network. Consequently, if the user is continuously typing the algorithm performs only a single notification packet transmission every NotifyTime.

**Algorithm 5** Typing notification

<table>
<thead>
<tr>
<th>Input: NotifyTime</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: <strong>Upon:</strong> text input area changed</td>
</tr>
<tr>
<td>2: <strong>if</strong> NotifyTimer = 0 <strong>then</strong></td>
</tr>
<tr>
<td>3: Transmit notification</td>
</tr>
<tr>
<td>4: NotifyTimer ← NotifyTime</td>
</tr>
<tr>
<td>5: <strong>end if</strong></td>
</tr>
</tbody>
</table>

When the other user receives the typing notification, it will show the other user the notification within a NotifyTime period. For example, we observe that WhatsApp uses approximately a 3 seconds timer for the NotifyTime.

### 6.3 Bundling of messages

Bundling aggregates consecutive messages sending them together, ideally in the same packet. The potential benefits are the following:

- **Less protocol overhead:** Since the small packets predominate the IM text traffic, sending the messages in the same payload instead of sending them in separate packets would reduce the amount of protocol overhead.

- **Extended idle time:** Reducing the number of transmissions by aggregating messages in the same bundle allows the UE to extend its idle time, and therefore the energy consumption.

- **Opportunity for better compression:** Compression techniques employ recurring patterns in the data to be compressed. The more data the better the compression result. We found that only 3% of the messages of our dataset would benefit from compression due to the short message size. We conjecture that aggregating messages would improve the compression results. The impact of compression is not studied in this work.

Even though the benefits of the bundle technique are convincing, the main drawback is that the technique delays messages in order to send them together. However, it is unclear whether introducing delay for consecutive
messages is detrimental for the QoE when the other user is not active in the conversation.

In order to investigate the potential energy savings of the bundle technique and the QoS in terms of message delay, we propose a simple event-based algorithm which uses information from the graphical user interface. Algorithm 6 describes the operation of the bundle technique. The intuition behind the algorithm is that when the user presses the send button the message is not directly sent over the network. Instead, a BundleTimer is started (or restarted if it is already running). The content of the text input area (i.e., the message $m$) is queued in the message queue $Q_M$.

**Algorithm 6 Message Bundling**

**Input:** $Q_M$ (initially $\emptyset$), BundleTime

1. **Upon:** send button pressed
2. $m \leftarrow$ text area content
3. $Q_M \leftarrow m$
4. BundleTimer $\leftarrow$ BundleTime
5. **Upon:** text input area changed
6. BundleTimer $\leftarrow$ BundleTime
7. **Upon:** BundleTimer = 0
8. if $Q_M \neq \emptyset$ then
9. Transmit $Q_M$
10. $Q_M \leftarrow \emptyset$
11. BundleTimer $\leftarrow$ BundleTime
12. **end if**

Whenever the text input area is changed, it means that the user is typing the next message. Therefore, the BundleTimer is restarted every time the user changes the text in order to bundle together the new message. This prevents the $Q_M$ to be transmitted while the user is typing, thus recognises consecutive messages.

However, if the user never stops typing, the messages will never be sent. We argue that if the user keeps on typing, the messages can be considered as the same one and send them together, no matter the length of the message. Therefore, we consider this case to be a pathological case.

The BundleTimer is a statically configured countdown clock, with the BundleTime input value. Whenever it expires, the messages in $Q_M$ are transmitted over the network, the $Q_M$ is emptied and the BundleTimer is restarted.

The proposed BundleTime parameter provides flexibility: a short BundleTime decreases the delay for single messages. A longer BundleTime would allow text over several consecutive messages to be aggregated. Since the text input area changed event requires the screen to be switched on, we do not consider more general indicators such as screen on or similar for simplicity.
6.4 Evaluation environment and methodology

This section describes the methodology and the evaluation environment used to quantify energy consumption of the typing notification and the bundle technique using EnergyBox.

The general methodology is as follows: a predefined set of real conversations is automatically replayed between a pair of clients using a prototype IM application. First, the energy consumption of each conversation is measured as a baseline. Then, the energy consumption including the typing notification is measured for the same conversations. Third, the saving achieved using the bundling mechanism is measured and compared against the baseline. The energy is calculated using EnergyBox from the resulting packet traces.

We briefly describe the implementation details of the algorithms and the employed environment as well as the evaluation conversations and EnergyBox parameters.

6.4.1 Prototype IM application

The client application provides the chat functionality between two Android devices and implements the logic of the bundle technique and the typing notification. There were three main reasons behind the decision to select to use the MQTT protocol as the transport protocol: it is a lightweight application protocol, some IM applications officially use it (e.g., Facebook Messenger), and the Mosquitto open source project provides easy to set up public MQTT servers that accelerate the development phase\(^2\).

![Architecture of the IM prototype implementation and the test environment.](image)

Figure 6.3: Architecture of the IM prototype implementation and the test environment.

\(^2\)Mosquitto test server: http://test.mosquitto.org/
Fig. 6.3 shows the general architecture of the application and the test environment. The IM clients are able to communicate using MQTT through the server that hosts the MQTT broker. Recall that MQTT is a publish/subscribe protocol, where the subscribers are instantaneously notified whenever a publisher generates a new event. Each user subscribes to her nickname. The other users send a message by publishing the message to the nickname (i.e., topic) of the recipient. The MQTT broker keeps track of the subscriptions to different topics. Since typing notification and bundle technique are client-based, the standard MQTT broker suffices.

The implementation of the bundle technique and the typing notification feature are mainly based on instances of the Async Task class provided by Android, where the Async Task represents the timers that can be cancelled, reset and started again. Both features are implemented at the Activity since they require to access to the user interface events. For typing notification, the task simply sleeps for the duration of the NotifyTimer. If the user pushes a button when the NotifyTimer is 0, the MQTT client part sends a special notify packet. The other user displays 'User is typing' message on the screen.

When the Async Task expires for the bundle technique, the messages held in the message queue are concatenated and delivered in one message to the MQTT client part. A user pressing a button while having the chat input field in focus restarts the Async Task.

6.4.2 Data set and parameter settings

This section describes the parameter settings and the conversations used to study the energy characteristics of the typing notification and the bundle technique.

Test conversations: Four different test conversations are selected for the tests. Dense and Sparse are conversation fragments selected from the IM dataset of section 6.1 in order to represent two different conversation patterns observed. These are used to describe a chat intensive conversation (short IMIs) and a slow-paced one (longer IMIs). Random is a test conversation generated by randomly selecting a set of messages from the dataset. The selection needs to follow the following rules: Only message pairs with an IMI less than 30s are selected since larger IMIs are uninteresting from the energy perspective (greater than the typical inactivity timers). Moreover, if a message that has consecutive messages from the same user is selected, the consecutive messages are also selected to keep the realistic message relations from the dataset. Finally, Short is the same synthetic conversation used in the background section 2.1 to compare the different IM applications. It is characterised for containing short messages, a single consecutive message, and IMIs are between the Sparse and Dense conversations.

The test conversations used for the tests are shown in Fig. 6.4. Note that the conversations do not start from 0 since the time to write the message is
also considered, and the difference in their length is not important since the results are studied per conversation.

![Figure 6.4: Test conversations.](image)

**Writing speed parameters:** In order simulate the user typing and automate the tests, the prototype application is instrumented with the test automation capability that simulates the user writing characters at a parametrised writing speed. A simplistic approach is adopted. For each message of a given conversation, the characters of the message are written using a constant typing speed. The typing automation logic starts writing in the input text field of the application so that the message is sent in its correct timestamp. The time to start writing the message is calculated using the message length and the average writing speed.

For simplicity, the typing speed is set to a constant of 287 ms delay per character. This number was experimentally obtained by measuring the time to write 160 characters (i.e., a SMS) for 6 different people.

**Correct order replay:** The test automation module employs 2 different techniques to write the messages of the test conversations based on their timestamp. *Exact timestamps* are used for the typing notification feature, i.e., the message is sent exactly at the timestamp. Since the bundle technique creates delays and could lead to out-of-order conversation, *reactive timestamps* are used to simulate how the users would write the conversation if the bundle technique would be active. The clients wait for each others’
messages in order to guarantee that all the messages are replayed in correct order.

The packet traces are captured in the device of the user 1 using `tcpdump` for every test, which are later used for computing the energy consumption. Only the traffic of the IM application is allowed using a firewall to block the rest of the traffic.

**EnergyBox parameter setting:** We set the 3G network settings that correspond to the operator TeliaSonera measured in our local area. The inactivity timers $T_1$ and $T_2$ are set to 4.1 s and 5.6 s respectively. The RLC buffer thresholds correspond to: $B_1^u = 1000$ and $B_2^u = 294$ bytes for uplink, and $B_1^d = 515$ and $B_2^d = 515$ bytes for downlink. The time to perform the different state transitions are set to 1.7 s, 0.65 s and 0.435 s for PCH-DCH, FACH-DCH and PCH-FACH respectively. The UE power values for the different RRC states are based on earlier measurements [95]: DCH = 612 mW, FACH = 416 mW. We set PCH = 0 W in order to quantify only the energy spent for transmission energy.

### 6.5 Energy cost of the typing notification

Aggressive notification policies can lead to an energy wasteful situation. This section studies the additional energy cost incurred by the typing notification functionality for the different test conversations. The results are based on 3 repetitions of each unique test. An additional 2 repetitions are run if great standard deviation is observed in the results.

For each conversation, we compute the energy consumption with the typing notification functionality disabled as a baseline. We select 3, 5 and 10 s as NotifyTime values for the different tests. These values are representative of the parameters used in real typing notification mechanisms (e.g., WhatsApp uses approximately 3 s as the NotifyTime).

Since the focus is on the additional cost, we normalise all the values to the average energy consumed by the baseline for each conversation (39.05, 32.87, 42.37 and 22.56 Joules for the Dense, Sparse, Random and Short respectively).

Fig. 6.5 shows that in general the additional energy cost of the typing notification feature is great, varying from 1.4 to 2 times the base energy consumption (40-104% more energy). Since the functionality sends packets whenever the user is typing, it keeps the 3G interface in an active state for almost the whole duration of the test. The results for each conversation are explained next.

For the Dense conversation, the energy consumption for the different NotifyTime values is similar. Even though a longer NotifyTime implies sending fewer notify packets, when the device enters a high energy RRC state, sending more data does not incur a higher energy consumption. Most of the energy consumption is due to resetting the inactivity timers, and thus the message pattern of the conversation greatly influences the consumed
energy. Since the conversation is dense (messages are close in time), the impact of the typing notification is not as high as in the Sparse conversation.

The additional energy cost for the Sparse conversation is higher than for the Dense. The Sparse conversation has periods of time where the device is in PCH between message sendings. However, the typing notification messages greatly reduces the idle time. Since the 3 and 5 s NotifyTimes are shorter than the inactivity timer $T_2$, this is reset before it has expired. When the NotifyTime is 10 s, $T_2$ expires which results in higher idle time. However, the energy cost is still 64% more than the baseline. The Sparse conversation is characterised by having slightly longer messages, and thus the time to type for the user is longer and more notify packets per message are sent.

The results for the different NotifyTimes for the Random and Short are similar to the Sparse conversation. The additional cost for a NotifyTime of 10 s decreases since it enables the device to stay in idle between messages consuming less than the 3 and 5 NotifyTimes. The Short and Random conversations contain mostly short messages. When NotifyTime is 10 s, less notify messages are sent due to the user spends less time typing the messages.

To sum up, the typing notification functionality results in a high additional cost for an IM application. According to our experiments, the energy consumption increased by 40-104% for the different NotifyTime values. Even
though a longer NotifyTime avoids excessive usage of the 3G interface, the cost is still high.

6.6 Does message bundling pay off?

Next, we quantify the energy savings that bundling can provide at the cost of an introduced delay due to aggregating messages.

6.6.1 Energy savings

Each test conversation is tested with the following BundleTimes: 1, 3, 5 and 7s. We employ the reactive timestamps described in section 6.4.2 to ensure correct order replay. The results are normalised to the base energy consumption with the bundle technique disabled (41.62, 32.36, 41.61, 22.13 for Dense, Sparse, Random and Short respectively). The results are based on 3 repetitions of each unique test. Additional repetitions are run when greater standard deviation is observed.

Figure 6.6: Normalised average energy and standard deviation of the bundle technique for the different conversations.

**Energy savings:** Fig. 6.6 shows the energy savings for the different test conversations. The bundle technique provides energy savings when the algorithm successfully performs at least one bundle. The energy savings vary between the conversations, and thus we describe them separately.
The energy savings due to bundling range from 27 to 43% for the Dense test conversation. Since it contains many consecutive messages and these are often close in time (i.e., the user starts writing soon after a previous message is sent), even a short BundleTime of 1 s can achieve 27% savings.

There are two types of unwanted traffic effects found in the packet traces which increase the standard deviation (e.g., BundleTimes of 5 and 7 s for Dense and Sparse). TCP retransmissions occur sometimes, but since they are performed a few milliseconds after the original transmission they do not influence the energy consumption substantially. However, packets of the Address Resolution Protocol (ARP) are also found, which impact the energy consumption since it consistently occur 5 seconds after a previous TCP packet. The varying number of ARP packets have a small but recognisable impact on the energy consumption. Thus we ran 2 extra repetitions of the tests affected by this issue.

The results for Sparse are different since the conversations were chosen to describe different cases. The bundle technique achieves 16% energy savings with a BundleTime of 1 s. However, increasing the timer value does not increase the energy savings. The IMI of the consecutive messages is long for the sparse conversation, thus the short BundleTime values do not manage to perform most of the possible bundles. This is further described later.

Regarding the Random conversation, the energy savings range from 24 to 42% for the different BundleTimes. The shortest timer provides again significant energy savings. Even though BundleTime of 3 and 5 s perform the same bundles, their energy varies due to the unwanted traffic mentioned above. The greatest energy savings are for the BundleTime of 7 s.

A single bundle is possible for the Short conversation. This is not achieved since the inter-message interval of the two consecutive messages is longer than the BundleTimes, and the message lengths are short (i.e., the typing time is short).

**Bundling performance:** Next, we analyse the performance of the bundle technique for the different BundleTime in terms of number of bundles and messages per bundle.

Table 6.1: Number of bundles and bundles per message for the Dense conversation.

<table>
<thead>
<tr>
<th>BundleTime (seconds)</th>
<th>Bundles</th>
<th>Messages per bundle</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>3, 3, 2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>4, 3, 2</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>5, 3, 3, 2</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>5, 3, 3, 2</td>
</tr>
<tr>
<td>Possible bundles</td>
<td>4</td>
<td>5, 4, 3, 2</td>
</tr>
</tbody>
</table>

Tables 6.1 and 6.2 show the performed bundles for the different BundleTimes using the Dense and Sparse conversations. For Dense, the BundleTime of 1 s performs 3 bundles for both users out of the 4 possible bundles.
Only one more message is aggregated for the 3 s BundleTime. All the possible bundles are performed when the BundleTime is increased to 5 s achieving the maximum energy savings. Similar results are obtained for 5 and 7 s since no additional messages are bundled even increasing the BundleTime.

For Sparse, the BundleTime of 1 s performs a single bundle out of the 3 possible bundles. The 3, 5 and 7 s BundleTimes only perform an additional bundle of 2 messages.

**Protocol overhead:** The bundle technique also provides less TCP/IP header overhead. The number of packets sent in the Random conversation with the bundle technique disabled were reduced from 40 to 16 when using a BundleTime of 7 s. Thus, the messages are sent in the same packet reducing the TCP/IP header overhead.

To sum up, our results show that even a short BundleTime can lead to significant energy savings. The next section studies the potential drawback of the bundle technique.

### 6.6.2 Message delay

Even though the bundle technique is desirable from the energy perspective, one needs to also consider its negative impact on per-message delay. The bundle technique delays each message by a minimum of the BundleTime value. The delay of the held messages increases when the user continues typing a consecutive message since the BundleTimer is restarted every time the user types a character.

The per-message delay is computed for the Dense and Sparse test conversations for the different BundleTime experiments. The application is instrumented in order to obtain the delay between the moment of pressing the sending button and the time that the message is actually sent to the network. The minimum delay for each BundleTime is the BundleTime itself, and it represents that a message was not bundled. The bundled messages are typically delayed more than the minimum, depending on the inter-message interval, the BundleTime and the character length of the next message.

Fig. 6.7 shows the delay introduced by the bundle technique for the Dense and Sparse conversations. The median delay for the Dense conversation with the BundleTimes 1 and 3 s is the BundleTime since most messages
are just delayed by the minimum delay. These are the last messages added to a bundle or the messages that were not bundled. As expected, the bundled messages experience greater delay. Fig. 6.7 shows that the maximum delay for the BundleTimes 1 and 3 s is up to 20 and 30 seconds. The maximum delay for 3 s BundleTime increases because it bundles an additional message. The more messages in the bundle, the higher is the maximum delay of the first bundled message. However, increasing the BundleTime leads to great increase on the per-message delay, especially the messages that are bundled. In Dense, even though the BundleTimes 5 and 7 s form exactly the same bundles, the messages experience and extra delay with no extra gain.

The results for the Sparse conversation show that the median delay is the minimum experienced delay. The long inter-message intervals of the Sparse conversation make the bundle technique to perform a single bundle of 3 messages for the 1 s BundleTime (the outliers of 6 and 11 s in Fig. 6.7), and an additional bundle of 2 messages for the rest. Thus, the per-message delay is shorter due to the smaller number of bundles than for the Dense conversation.

Comparing the energy savings and the introduced per-message delay, we observe that a short BundleTime of 1 s leads to significant energy savings while not causing in great delay. This indicates that simply keeping track of the user typing is enough for a simple bundle policy. Longer BundleTime values can increase the energy savings for dense conversations at the cost of higher delay. However, Sparse conversations with sporadic messages should not use long timers.

Finally, the message delay does not always have a negative impact for the QoE. When the user is not engaged in an active conversation, the reception of the non-delayed messages or a bundle of delayed messages can be argued to be the same. However, from the energy perspective, the latter reduces drastically the energy consumption.
6.7 Summary and discussion

When developing energy-efficient solutions for application data transmission, there is a trade-off between the adding application functionalities that perform network transmissions and energy conservation. There is a need to quantify the extra energy cost and the perceived functionality benefit from the user side. EnergyBox is a valuable instrument for this purpose.

In this chapter we quantified the typing notification functionality employed by most IM applications and showed its tremendous energy cost. According to our results, this functionality can increase the energy consumption of an IM application by 40-104% from the basic message exchange functionality. Quantifying the cost of a functionality can allow the developers to rethink the need of the functionality or provide the option to dynamically enable/disable it.

When the traffic is directly generated by user interaction, the network transmissions can easily result in energy waste. The knowledge of the typical usage of the application can be used to reveal energy inefficiencies and design tailor-made energy-efficient solutions for a type of application. Based on our small scale study with collected real usage data, we conjecture that IM application users tend to write consecutive messages which increases the active time of the wireless interface. Thus, we believe that bundling can be used to reduce transmissions while the user continues typing, and send them at once.

The bundle technique was observed to result in energy savings up to 43% depending on the message pattern of the conversation. Given the high percentage of the consecutive messages, this is a promising result. However, using longer timers for the bundle technique can lead to high delays for some messages. Thus, the bundle technique should be dynamically activated when the user is offline or away. This could be automatically deactivated in periods of high intensity conversations, i.e., for real-time communication in which responsive requirements are high.
Chapter 7

Related Works

Recent years have seen a growing interest in measuring and reducing the energy consumption in mobile devices towards green networking. Mobile network operators have become interested in the energy saving approaches as well, mostly due to increasing cost of their networks [34]. However, new proposals often adopt infrastructure-centric approach rather than considering a holistic approach including user terminals [49,69,70,75,126]. A comprehensive description of the challenges of green cellular networks is provided on a survey by Hasan et al. [42]. This chapter provides a survey of the related work focused on energy consumption of data transmission at the user end, and thus infrastructure-centric approaches are not considered. The same reasoning is applied also for other technologies such as WiFi [51,67,102].

The general categories of the taxonomy were briefly described in section 2.3 and the taxonomy is shown in Fig. 7.1. The work presented in the different chapters is within the highlighted categories.

Some works appear in more than one section due to their different contributions (e.g., measurement study and solution proposal).

7.1 Measurement studies

Measurement studies typically aim at revealing inefficiencies, discovering the footprint characteristics of wireless interfaces and providing guidelines for energy efficient operation. Experimental setups for energy consumption studies on mobile devices are characterised for using either built-in software in mobile devices like Nokia Energy Profiler (NEP) [18,39,80,83,89,111] or external power meters [36,95,99,104,113].

We categorise the different works based on the technology used in their study. Some works focus on various technologies in order to compare them.

**Cellular communication:** An influential measurement study by Balasubramanian et al. [18] reports the categorisation of three different energy components in cellular networks: ramp, transfer and tail. The tail compo-
Energy-efficient wireless networking at the user end

- Measurement studies
  - Ch. 5 and 6
- Energy saving techniques
  - Analytical
  - Measurement-based
  - Ch. 3
- Link selection algorithms
  - Wake up
  - General selection
  - Size-based
  - Time-based
  - Ch. 4 and 6
- Low-level power saving
- Traffic adaptation
- Cooperation techniques

Figure 7.1: Taxonomy of wireless transmission energy at the user end.

...ent is the most significant and is caused by the inactivity timer statically set by the network operators. Our previous work [13] refines this study by performing physical measurements using a cellular modem that isolates the energy for data transfers. Their work uses bulk data transfers, which do not characterise the impact of the data pattern on the energy consumption.

A detailed study of the tail energy overhead in 3G cellular networks is performed by Qian et al. [95] on user data traces retrieved from a network operator in 2009. The work is extended [94] pointing out how different applications inefficiently utilise the radio resources due to their data pattern. Both works provide an excellent ground for understanding the impact of the operator-configured timers and RLC data buffer thresholds on the theoretical user battery discharge.

Schulman et al. [104] explore the impact of the network coverage on the energy consumption showing that mobile devices consume more energy when sending data under poor link conditions.

Puustinen et al. [89] measure the impact of unwanted Internet traffic on the energy consumption for 3G and propose the usage of firewalls to block the traffic and Fast Dormancy as a potential solution.

Perrucci et al. [85] focus their study on the energy consumption difference of 2G and 3G according to common usage of the networks, such as SMS, voice calls and data traffic. They suggest to use SMS compression for long messages, which could be also applicable to other type of data. The study shows that the energy per bit of 3G is significantly smaller, suggesting a intelligent vertical handover between 2G and 3G depending on the usage pattern and considering the energy cost of the handoff. Our previous work...
[13] confirms the benefit of using 2G instead of 3G for small amounts of data.

The main advantage of our measurement setup is that we are able to isolate the energy consumption of data transfers by performing measurements in a 2G/3G/GPS module.

The benefit of using EnergyBox to efficiently complement physical measurements is shown in Chapters 5 and 6. Chapter 5 shows significant difference in energy consumption for different application design choices in terms of protocols and data format.

The related work in the context of IM applications is focused on reducing the presence [29, 71]. Chapter 6 extends the work studying the high cost of using a notification mechanism by applications and the great energy consumption difference between IM applications.

**WiFi communication:** Rice et al. [99] measure the WiFi energy consumption in a variety of smartphones. Their experimental methodology for measuring mobile devices is an inspiration in our work. Using their framework they study the energy consumption of the DHCP process and data transmission in 802.11 networks in different devices, reaching the conclusion that mobile devices would make the most out of a transmission with big message sizes. Their work reports the power consumption differences due to hardware and software generations.

Several works focus on studying the energy consumption of WiFi based on bulk data transmissions [18, 38]. Studies using bulk data transmissions do not capture the impact of adaptive PSM on energy consumption. Various works [67, 91, 102] study the different adaptive PSM mechanisms in previous generation devices. A recent study by Pyles et al. [90] shows the threshold to switch between PSM and CAM for different devices.

Atheros [14] studies the power consumption of various 802.11 based WLAN cards reporting that the energy consumption decreases when increasing the data rate due to shorter transmission and reception times.

Regarding the ad hoc mode, a fine-grained study of a IEEE 802.11b wireless network interface is performed by Feeney et al. [36] showing that apart from sending, receiving and discarding traffic also increases the energy consumption. They report a high energy cost of the idle state and that big packets are more efficient. A more recent study of the energy consumption and data rate of the IEEE 802.11b/g ad hoc mode is performed by Pedersen et al. [80] placing in line 16 smartphones equally spaced with 3 meter of distance. The work reports how bigger packet sizes lead to less energy per byte and increases the MAC layer efficiency.

**Bluetooth:** A detailed study of the Bluetooth 2.0 energy consumption is performed by Ekström et al. [32] showing that the energy consumption of the master mode is proportional to the number of slaves, increasing substantially per additional slave. Moreover, their work shows that the transmission power does not increase the overall energy consumption (only small peaks).

The energy consumption of the different link layer packets of Bluetooth
using the asynchronous connectionless link (ACL) is studied [87], showing that using packets without error correction scheme (DH packets) is better in terms of energy consumption and throughput.

**Comparative studies:** Friedman et al. [38] present a comparative study between WiFi (AP and ad hoc modes) and Bluetooth in terms of throughput and power. Their measurements of the aggregated energy include CPU and memory costs.

Wang et al. [114] present the energy consumption of bulk data transfers (transfer a given size of data at once) over WLAN and cellular networks. The study show that mobile devices would benefit from big packet sizes, high data rates and bursty transmissions to allow longer low power consumption periods in the interfaces.

Xiao et al. [116] show that the download of videos in WLAN is more energy efficient than in 3G, suggesting a tailor-made solution for streaming using WLAN.

PowerScope [37] is a tool for profiling energy usage of computer applications using hardware instrumentation. Their work maps the sample data of the to profile the energy usage by process and procedure employing kernel level system activity information. Using their system they significantly reduce the energy consumption of an adaptive video application.

Several other measurement-based energy consumption studies focused on mobile devices are not considered here due to their non-networking related nature. The readers interested in other energy aspects than communication for single mobile devices are referred to the survey by Vallina-Rodriguez et al. [110].

### 7.2 Energy saving techniques

Energy saving techniques are categorised in different groups based on their aim and the means used to reduce the energy consumption. The solutions presented in Chapters 4, 5 and 6 are categorised under traffic adaptation techniques and are compared against the other works.

#### 7.2.1 Link selection techniques

Link selection techniques are based on the intuition that some interfaces (e.g. WiFi, 3G, Bluetooth, etc) are more efficient than others given factors like power consumption of the interface, network conditions, link availability or amount of traffic and traffic pattern. These techniques aim at selecting the most energy efficient link at a certain point in time. Switching between different types of interfaces is referred to as vertical handover or handoff.

The most basic algorithms are wake-up techniques. These simply employ a low power interface during idle time and wake up the high power interface when there is any data transmission. The general selection techniques attempt at dynamically select the best available link at a given point.
in time. Different technologies and mechanisms are considered to estimate the current network conditions to select the interfaces.

**Wake up techniques:** There exist some literature presenting the possibility of using a low power interface to wake up a second one, assuming that the mobile device is always connected via some wireless interface. Perruci et al. [83] wake up the WLAN interface in order to receive VoIP traffic using cellular-based wake-up signals or collaborative approach, where another device connected to WLAN wakes up the intended receiver. A similar work by Agarwal et al. [7] aims to reduce the idle cost of WLAN for VoIP traffic using the cellular network to notify and wake up the WLAN interface. Their approach is evaluated showing significant energy savings. A general approach to reduce the idle energy consumption of WLAN using a low-power radio to wake up the principal one is proposed by Shih et al. [106]. These works only utilise a wireless interface for waking up another interface that is always assumed to be more energy-efficient.

**General selection:** General selection techniques consider the usage of all the wireless interfaces. Coolspots [82] enables automatic link selection between Bluetooth and WLAN assuming that both interfaces are always available. The mobile device switches up from the low-power to high-power interface measuring the channel capacity via round-trip time, and switches down based in a bandwidth threshold set by measuring bandwidth levels that previously triggered the interface switch. Their work is based on the assumption that the access point implements both interfaces, which is not usually the case.

Vertical handoffs between cellular and WLAN have attracted great attention in the literature. Nam et al. [73] propose WISE, an algorithm that selects between 3G and WLAN based on the characteristics of the interfaces and the current data traffic.

Balasubramanian et al. [17] propose Wiffler based on a detailed study of 3G and WiFi availability from 3 cities. Wiffler attempts to reduce the cellular traffic by delaying transfers to offload as most data as possible to WiFi and switching back to 3G when WiFi is unable to transmit the data within a small time window.

Ra et al. [96] propose and implement an algorithm developed using the Lyapunov optimisation framework, that based on power cost of the interface, transfer rate estimation and queue backlog determines when and which link to use for bandwidth-intensive delay tolerant applications (e.g., video upload). The transfer rate is estimated combining an online and offline methods based on measurement and using the Received Signal Strength Indicator.

Intentional Networking [44] makes developers to provide semantic labels (e.g., small or large and foreground or background) to differentiate the data transmissions and match them to the different interfaces (WiFi or 3G). Even though the aim of their work is on optimising the QoS rather than energy, the results of opportunistically deferring messages to send it via the selected
interface would have a benefit on the energy (similar to traffic adaptation techniques). However, their proposal requires developers to describe via the labels their transmissions, and therefore rewrite parts of the applications.

Depending on their context, some link selection techniques focus on proposing vertical handovers using the Media Independent Handover standard, whereas others focus on improving the estimation of the different network parameters that can impact energy or performance:

- **Media Independent Handover**: Although it is not specifically focused on reducing energy consumption, the IEEE 802.21 Media Independent Handover (MIH) [50] facilitates vertical handovers that can be initiated by either the network or the mobile device. MIH aims to provide and support inter-technology handover ensuring the performance in terms of latency and loss and helps with the handover initiation and handover preparation (network selection and interface activation). However it does not standardise the handover execution mechanism. In this context, Taniuchi et al. [108] present an introduction to IEEE 802.21 describing the elements of the specification and their own handover implementation between cellular (EVDO) and WiFi based on signal strength. Rodriguez et al. [100] propose a middleware architecture to support vertical handovers with MIH using signal-to-noise ratio and QoS parameters for the handover decision. A similar approach is presented by Kim et al. [53] without including energy consumption of the different links. Rahmati et al. [97] address the TCP/IP flow migration when performing a horizontal (i.e., same technology) or vertical handover.

- **Measuring network availability**: Rahmati et al. [98] statistically model the WLAN availability using conditional probability distributions of network conditions given the context information in order to determine the expected energy savings of performing the vertical handoff. Their work proposes 4 algorithms to estimate WLAN network conditions with the context information. Footprint [115] propose a cellular positioning based algorithm to reduce the energy consumption of WLAN by reducing the number of scans. Even though they do not explicitly propose the vertical handover, their work covers the fundamental WLAN access point discovery part to proceed to perform the handoff.

In general, the possible energy benefit of doing a vertical handover is considered as a desired side effect due to the main goal of these works is on improving the performance of the handover rather than reducing the energy consumption.
7.2.2 Low-level power saving

Several works aim at improving the low-level power saving mechanisms provided by the standards by tuning parameters for different types of traffic.

**3G:** Fast Dormancy (FD) is the mechanism introduced in the 3GPP Release 8 that allows the UE to signal the desire to move to PCH before the expiration of the inactivity timers. Some proposals [15,30,93] employ different flavours of FD to reduce the energy overhead of the inactivity timers. These works focus on 3G periodic transfers and background traffic. While current mobile devices use fixed FD timers to release the channel, the above works aim at mining application information to reduce energy while not increasing the signalling overhead and the state transitions (i.e., latency). Yang et al. [120] study the parameters of the power saving discontinuous reception (DRX) mechanism performed between the network and the mobile station.

Several works aim at optimising the values of the inactivity timers to either reduce the energy consumption of the handsets while still maintaining network performance or reduce channel occupancy [63,64,95]. The different proposals range from global dynamic timers to fixed timers per user.

**WiFi:** Several works improve the standard power saving mechanisms of WiFi. The concept used in current devices (adaptive PSM) was introduced by Krashinsky et al. [58]. The sleep durations of the PSM are modified given the traffic levels while maintaining a level of RTT. Anand et al. [9] propose to adapt the PSM behaviour given the current traffic and the application intention provided by the developers. Pyles et al. [90] assign priority labels to each application using a machine learning classifier and the switch between PSM and CAM is triggered only by high priority applications. They observe significant energy savings evaluating their adaptive PSM implementation.

While improving the power saving mechanisms is a generic technique, the transmissions of an inefficient application would still waste energy even with an optimised power saving mechanism.

7.2.3 Traffic adaptation techniques

Given the footprint characteristics of a wireless interfaces (e.g., data rate or power saving mechanism), energy-aware traffic adaptation techniques aim at adapting regular traffic to create energy-efficient traffic. The traffic adaptation is usually performed by shaping the traffic over time, reducing the amount of data sent or a mix of both by altering the data pattern using different techniques. 3G periodic transfers and background traffic of mobile applications are known to incur a great energy consumption. There exists a vast literature aiming to reduce its energy consumption. Many techniques accommodate different flavours of FD within their proposed technique due to it is an improvement over the standard, and orthogonal to traffic adaptation techniques.

**Size-based:** Some techniques consider the benefit of sending less data or
stripping the redundant data (e.g., compression). The viability of data compression for wireless communication in mobile devices is studied by Wang et al. [113] concluding that it can provide up to 50% of energy savings by reducing the amount of data transmitted. Stratus [8] combines compression with bursty communication during good signal strength periods. Armstrong et al. [12] propose an infrastructure to support dynamic content updates by only updating the interesting part within Web pages selected based on the client interest reducing the transmission data.

While compression works for big amounts of data, the work presented in Chapter 6 in the area of IM applications has shown that small messages do not benefit from compression. These would require a combination of size and time techniques to compress several messages at the same time.

**Time-based:** Given that data transmissions can be shifted over time, different techniques enable to create energy efficient traffic by aggregating transmissions.

- **Traffic aggregation:** Traffic aggregation defines the set of techniques that defer the data transmissions within a maximum period until one of them needs to be sent (e.g., user-specified deadline). All the transfers that were postponed until the last moment are sent together reducing the number of state transitions and energy tails at the cost of the delay.

TailEnder [18] employs an online scheduling algorithm using user-specified deadlines, and predicts future data accesses to prefetch data to reduce network transfers. Each packet is assigned a deadline and buffered. When the deadline of any of the packets is about to expire, all the buffer is sent in burst. CLBB (chapter 4) schedules data transfers considering the inactivity timers and the RLC data buffers in combination. Our evaluation shows an improvement over TailEnder.

In a similar way to TailEnder, TailTheft [65] combines traffic aggregation and FD. Their work shows an improvement in prefetching the content by performing more accurate usage predictions.

Bartendr [104] schedules bulk data transmissions in periods of good signal strength. Nguyen et al. [74] improve the network condition predictions using a mixture of location, time and signal strength to schedule data transfers. Their work is orthogonal to ours since CLBB could benefit from their network estimations to alter the transmission based on the link quality.

Zhang et al. [125] propose different algorithms that aim at maximising the battery lifetime given a battery capacity and different workload considerations using a simplified model of the 3G interface. An optimal algorithm is proposed that transmits the maximum amount of data given the battery capacity. Their work does not consider the inactivity timers or RLC buffer thresholds that make the data pattern drastically impact the energy consumption.
7.2. ENERGY SAVING TECHNIQUES

At application level, the bundle technique proposed for IM applications in Chapter 6 follows the same idea of the above works by aggregating messages. However, we employ user activity information to perform a more efficient message aggregation.

Cinder [101] is an operating system that aims at providing fine grained power management for mobile devices. Processes are provided with a limited energy budget (reserve) and a energy discharge limits (taps). The high cost of sending data through the cellular interfaces leads to traffic aggregation since a single process has not enough energy budget to perform the transmission alone. When a process wants to send something, its packets are deferred until there is enough energy budget to perform the transmission.

Based on the energy-delay tradeoff, the same concepts can be also applied to other technologies. For example, Palit et al. [77] employ packet aggregation applied to WiFi, where packets are aggregated at the medium access control layer in order to fill a full frame and allow the device to extend its doze time. However, the timescale is in order of milliseconds whereas cellular works consider seconds.

- **Batching:** Batching refers to grouping several periodic transfers into a single one, i.e., increasing the period of the transmissions. The batching point is usually defined in advance and the applications send their traffic in those time points. Batching and traffic aggregation can lead to different results. The batching points are periodic and non-traffic dependent. However, traffic aggregation techniques usually consider the traffic.

Fig. 7.2 shows the conceptual difference between traffic aggregation and batching. For traffic aggregation, each packet is delayed up to a maximum delay and buffered. The packets are sent in burst before the maximum delay. The batching points are defined in advance given a batching interval. Then, the packets are shifted in time and sent in those batching points. The example shows how the same time for batching interval and maximum delay can lead to different bursts due to traffic aggregation depends on the packet arrival time.

Könönen et al. [57] propose the alignment of timers of different applications to perform synchronised bursty transmissions with a determined periodicity. Calder et al. [24] schedule data transmissions of applications with different transmission intervals in multiples of the shortest interval.

Huang et. al [48] show the energy consumption of the traffic generated when the screen of the device is off in 3G and 4G networks, and propose a combination of FD and batching to reduce background energy consumption.
• **Piggybacking**: Finally, Piggybacking is a method where background data transfers can be deferred or moved earlier in time to match the transfers with users’ foreground traffic. Traffic backfilling [59] uses piggybacking to opportunistically transmit background data during unused gaps of energy tails between interactive traffic. This can be considered as opportunistic batching.

CLBB considers the current RRC state of the UE to schedule the background data. If background and foreground traffic differentiation is possible, the foreground traffic can alter the RRC state making CLBB to perform piggybacking, i.e., transmitting background data together with the foreground data. However our work does not present any evaluation in this regard.

Qian et al. [92] perform a simulation comparison of different approaches (FD, piggybacking, traffic aggregation and batching) showing that a combination of approaches can lead to higher savings.

Compared to the above works, CLBB is the only algorithm considering the RLC buffer thresholds when scheduling data transmissions to avoid unnecessary state transitions.

### 7.2.4 Cooperation techniques

Different techniques are analysed in the literature in order to reduce the energy consumption of the terminals when considering several nodes at the
same time. Cooperation techniques combine different wireless interfaces with different characteristics and often employ node clustering techniques. The energy savings are mostly achieved by using a less energy consuming interface in a similar way to link selection techniques. However, the focus of cooperation techniques is on creating the availability of the less energy consuming option by connecting among several nodes. The aim of the cooperation techniques is commonly divided in two cases: minimise the aggregated energy consumption of all the devices (global) or minimise locally the energy consumption of each terminal (local).

Ad-hoc: While forming ad hoc configurations as opposed to using an existing infrastructure is a widely studied topic [28], energy consumption is known to be the Achilles’ heel of ad hoc communication [109,111].

Clustering: The ground of these techniques is on the Wireless Sensor Networks (WSN) field. Compared to the WSN literature [10, 23, 43], in cooperation techniques there is no common goal for the whole network (e.g., data collection), and the footprint characteristics of the different interfaces and the traffic load for the individual nodes is different and unpredictable.

Cooperative approaches using Bluetooth and WLAN [122], or cellular and short-range interfaces such as Bluetooth or WLAN [84, 86] typically lead to great energy savings due to the great energy consumption difference between the interfaces.

Software access point: Recent mobile devices are able to set up on-the-fly software access points (SoftAP) leading to clustering. Several works aim at reducing the cost of using the SoftAP by improving the power management techniques [25, 40, 52] or offloading the energy burden from the mobile to a laptop that becomes the SoftAP [105].

Cognitive radio: Several works consider the dynamic intelligent radio to reduce the energy consumption as well as increase performance. The interested reader is directed to the available surveys [68,112].

Reduce energy by increasing performance: Some works [39, 66] focus on improving the performance in terms of bandwidth. This is done allowing the nodes experiencing good link quality relay the traffic from the poorly connected ones. This results in saving energy by shortening the transmission times and improving the media access.

While the energy saving potential of cooperation techniques is promising, security and privacy issues are commonly the main impediments that hold back the applicability of the techniques. The fairness between the different users and the incentives for the users to form part of the cooperation techniques are also significant problems to solve. In this context, Lei et al. [61] analyse the business models in the case of operator controlled peer-to-peer communication in combination with Long Term Evolution.


7.3 Energy models

Energy models abstract the real behaviour of the devices characterising the mechanism that consume energy. Some works focus on specific mechanisms whereas others attempt to model the total energy consumption. Energy models are typically either measurement-based or analytical. Since EnergyBox is our measurement-based proposal in this regard (presented in Chapter 3), we comment on the main works related to our approach.

Measurement-based models: Balasubramanian et al. [18] model the energy consumption of 3G and WiFi using linear regression based on their bulk data download measurements. Given the amount of data to be sent in burst, their model outputs the energy consumed. Their simple model does not capture the impact of the data pattern on the energy consumption and it is dependent on their measurements (e.g., device and network settings).

Oliveira et al. [76] use a state based model for 3G modelling the RLC buffer occupancy as fixed data rate (kbps) without performing any validation of their model.

Qian et al. [95] propose a finite state machine 3G model to study the impact of the inactivity timers on the energy consumption using a large traffic dataset. The radio resource usage application profiler for 3G presented by Qian et al. [94] is the work closest to our EnergyBox for 3G, which extends their previous model. We extend their 3G work by including in our 3G state-based model a low activity mechanism implemented by some operators, thereby improving the accuracy of the model. Moreover, our EnergyBox captures different RRC state machines within a single parametrised one.

The work by Harjula et al. [41] creates a device-specific power profile for different messaging intervals based on device measurements on a Nokia 95 for a single operator. Their model uses indicators such as average packet size and signalling frequency that need to be extracted from the data transmissions. Instead, EnergyBox directly works on data traces and derives the RRC states of different networks simulating the RLC buffer thresholds and state transitions. Our approach is more general and simplifies energy consumption studies.

Xiao et al. [117] present a detailed power level model for the 802.11g interface. The modelled data pattern based on traffic burstiness was only validated by simulating TCP download and upload traffic based on the amount of data transferred. In comparison, our adaptive PSM model is validated against a set of representative application traces.

PowerProf [56] is an automatic power profiling method. A set of training measurements is collected to build the models by exercising the different components of a smartphone. The resulting models are training data dependent, which are likely to miss the underlying mechanisms of cellular and WiFi interfaces. Their method requires the definition of a wide set of relevant tests to exercise the different components. Instead, EnergyBox di-
7.3. ENERGY MODELS

rectly models the lower layer operation of 3G and WiFi. A combination of both methods would provide versatility and accuracy by merging automatic model generation and expert knowledge.

Several works aim at quantifying the total energy expenditure of a device considering the wireless interfaces as well as other components such as CPU, screen or audio interfaces. A recent study by Pathak et al. [78] present a system-call-based power modelling for smartphones, including device dependent 3G and WiFi models. Tracing system calls typically requires modifications to the operating system (e.g., root access), whereas we adopt a generic approach based on packet captures, which are generally available\(^1\).

The work by Mittal et al. [72] contributes with an AMOLED display model as well as other component models taken from the literature to propose an energy simulation tool for Windows phones.

Zhang et al. [124] propose an online energy estimation and model generator implemented on Android. Hardware components are trained to generate device specific models using system variables (e.g., LCD brightness level). For 3G, a single operator is modelled by sending packets based on a previous methodology [95]. The WiFi model is based on a finite state machine using the number of packets sent and received per second, and uplink channel and data rate. Their work proposes to substitute the external power measurements by the battery discharge curve and the current battery voltage. Since the battery discharge curve also varies from battery to battery, their approach requires to characterise the battery discharge for each battery separately.

Dong et al. [31] propose to automatically self-generate a system energy model within a device using the smart battery interface. Similarly to energy modelling in desktop computers, their work employs system statistics and memory usage and complements them with the battery interface readings. However, smart battery interfaces that provide current measurements are not available in most of the devices.

Huang et al. [48] compares the power and throughput characteristics for WiFi, 3G and LTE using measurement-based models for a single phone. The models include a linear throughput-power curve measured in the phone since the LTE power substantially increases with the data rate. A comprehensive explanation of the 4G (WiMAX and LTE) power saving techniques is provided by Kim et al. [54].

Ekström et al. [32] develop an energy model for Bluetooth 2.0 + enhanced data rate (EDR) focused on low-duty-cycle network applications based on measurements in Bluetooth modules.

**Analytical models:** Analytical models often model the behaviour of a specific mechanisms of a wireless interface and often suggest guidelines to select parameters of possible optimisations.

Zanella et al. [6] propose a mathematical framework for analysing the

\(^1\)Android phones provide traffic statistics that can be captured every millisecond and packet level capture is available without root permission from Android 4.0 on.
energy efficiency of IEEE 802.11 distributed coordination function (DCF) modelling statistically collisions of a cluster of n nodes. Their study shows that sensing and discarding traffic have big impact in the energy consumption of the terminals.

Zanella [123] also presents a mathematical framework for analysing the performance of asynchronous connection-less (ACL) Bluetooth links in terms of goodput, energy efficiency, delay and system lifetime under the assumption of heavy traffic. Applying the framework the best transmission frames can be selected depending on the signal-to-noise ratio.

Lee et al. [60] present analytical models based on modelled traffic behaviour (web and streaming) to compare the the impact of inactivity timers on the energy consumption. Using the same modelled traffic, Yeh et al. [121] analytically model the state transition behaviour using a discrete-time Markov chain to model the inactivity timers.

Yang et al. [120] analytically model the discontinuous reception mechanism cycle (DRX) of UMTS with bursty packet data traffic by a semi-Markov process. They provide guidelines to select the inactivity timer and DRX cycle value, two parameters that determine the power cycle of mobile devices in cellular networks. Yang et al. [119] also present an analytical model that is studied by simulation providing some guidelines in order to design dual mode (i.e., multiple interfaces) mobile devices based on UMTS and WLAN links.

The named works are limited to modelling web and streaming traffic, whereas the EnergyBox is applicable to arbitrary data traces captured from any real application.

The increasing number of works considering transmission energy at the user end shows the importance of an area with a very active community.
Chapter 8

Conclusion and Future Work

This chapter presents the general conclusions based on the performed work and discusses the potential future work directions.

8.1 Conclusions

Energy has become a scarce and valuable resource for mobile devices, which calls for its optimal utilisation. Wireless interfaces account for a great part of the total consumed energy, and this is forecasted to increase given the energy-hungry interfaces and the extreme growth of application data usage.

Quantifying the energy of data transmission can reveal hidden inefficiencies and lead to low energy footprint application designs. Our work proposes EnergyBox, a parametrised tool that accurately estimates the energy consumption of data transmission at the user end using packet traces as input. The users of the tool can quantify the transmission energy covering a wide range of scenario setups, and efficiently complement physical power measurements. EnergyBox facilitates the process of revealing inefficiencies based on the data pattern.

In this thesis, the development of energy-efficient solutions is based on two different approaches: Tailor-made energy-efficient solutions are conceivable for certain application types given knowledge about their operational data. However, minimising the energy consumption for a an entire traffic class requires a more generic approach. Our work has shown that EnergyBox becomes a valuable instrument in order to quantify the energy of both type of solutions in an efficient manner.

Tailor-made solutions: The impact of different design choices need to be systematically studied when developing the data transmissions of an application. The usefulness of EnergyBox as a tool to facilitate the developers
at the design stage to develop tailor-made solutions is shown in two different application contexts: location sharing applications and instant messaging applications. Our results show that the correct design choice in terms of application protocol or data format choice can substantially reduce the energy consumption for location sharing applications. For instant messaging, we show that there is a tradeoff between adding application functionalities that perform network transmissions and energy conservation. Bundling messages is shown to lead to a reasonable delay in the message delivery that can significantly reduce the energy consumption.

In general, tailor-made solutions lead to energy-efficient data transmission designs for applications. However, the main limitation of this approach is the uncertainty of the aggregated energy-efficiency of the system when all the efficient applications run in parallel. Without any coordination, the chances are that the different network transmissions will lead to an inefficient operation at the handset level. Nevertheless, given that the applications are optimised for lower energy usage, the aggregated energy consumption is foreseen to be lower than running inefficient applications.

**Generic solutions per traffic class:** The advantage of the traffic class solutions is the capability to consider the traffic from several applications at once. In our work, we perform a first step towards a generic solution employing the background traffic. The background traffic is characterised by having no QoS requirements, thus the data pattern can be adapted to create energy-efficient traffic. Since traffic class solutions operate below the application layer, we employ the knowledge of the hidden energy footprint characteristics of the radio communication layer to optimise the usage of the battery resource at the cost of delay. Moreover, we study the cost of running the solution itself and show that the proposed cross-layer background traffic scheduler uses small amount of energy to significantly extend the battery lifetime. However, we show the such solution requires application traffic differentiation.

### 8.2 Future work

Energy efficiency of wireless transmission at the user end is a very active and a significantly large area of research. While our work focused only on parts of the whole area, there are many research directions. We list some interesting future directions, ranging from immediate work based on the work of this thesis to more general future work.

- **Towards generic solutions:** Our work treated a single traffic class separately. The coordination of the different traffic classes generated from different applications is also challenging due to its different QoS requirements. Traffic differentiation becomes the main limitation of the traffic class approach. Application level traffic differentiation would require the developers to label their transmissions. Lower layer
traffic differentiation without application cooperation is an interesting approach to investigate. Future traffic estimation per-application can be a significant aid to optimise the scheduling of data transmissions.

- **IM applications:** We believe that IM applications are interesting to study for their wide adoption, and their different types of traffic that range from simple text messaging to multimedia transfers or VoIP. Our current work can be extended including multimedia messages and group chats. The bundle technique appears very interesting for group chats which are usually denser conversations. Employing user activity (e.g., the screen is off, the user switched to another application), context information (e.g., built-in sensors) or presence information from other users would benefit to improve the energy efficiency of this applications.

- **Automatic energy evaluation framework:** An automatic energy evaluation framework for applications is envisioned as an immediate extension of the EnergyBox. The development of methodologies and metrics to evaluate the efficiency of a single application still remains as an open question. How can we define energy efficiency classes? Which usage scenarios are representative for the evaluation? While tailor-made solutions might appeal as a feasible approach for developers, the time spent to optimise the energy usage of their application can be detrimental to the time to market and other priorities (e.g., bug fixes or development of new functionalities). We believe that a semi-supervised application testing approach that can automatically generate the user inputs can drastically facilitate and accelerate the energy evaluation.

- **Limited resources per application:** Assigning each application a limited energy budget to perform network transmissions would restrict the bad practices and wasteful energy behaviours. A game theoretical approach can be devised to model the applications as players, which want to minimise their energy expenditure given their network transmissions and their QoS requirements.

- **Energy as part of QoS:** Since energy is not currently considered as one of the parameters of QoS, there is a need to focus the efforts on developing methods and metrics for it. Different QoS classes can be defined based on all the parameters. This would allow to define energy requirements (e.g., maximum delay and maximum energy consumption).

- **Collaborative techniques:** Similarly to resource consolidation, consolidating the traffic of several nodes into few cellular links is an interesting future direction. The usage of a second interface could drastically reduce the total energy consumption of all the nodes.
• **Holistic infrastructure and user terminal approaches:** While the research in green networking focuses either on the infrastructure or the user side, approaches considering both ends could lead to more efficient solutions. For example, user transmissions can be scheduled to enable sleeping techniques for infrastructure nodes.

While the potential directions can significantly help to reduce the transmission energy consumption at the user end, there is still plenty of work to be done to achieve energy-efficient software and wireless transmissions.
Appendices
Appendix A

Energy Consumption of Recharging Smartphones

This Appendix estimates the amount of energy used worldwide to power up the current number of smartphones.

We selected two of the last generation smartphones (iPhone 5 and Samsung Galaxy S3) and obtain its time to charge from 0% to 100%, and the average power drain during the recharge period. The iPhone 5 and Samsung Galaxy S3 are shipped with 1440 mAh and 2100 mAh batteries respectively.

The Galaxy S3 consumed 12.3 Wh, whereas the iPhone 5 consumed 9.5 Wh. Note that the tendency in the manufacturers is to increase the battery capacity, where the recharging time will increase.

This estimation uses a simplifying assumption that an average smartphone user charges her device once per day. The year consumption is obtained by multiplying the consumption by 365 days: 4.49 kWh and 3.47 kWh for the Galaxy S3 and iPhone 5 respectively. For simplicity, we consider 4 kWh as the year consumption for every smartphone (10.95 Wh per day).

Given a billion of smartphones shipped around the world in 2013 and 4 kWh year consumption, the total cost of recharging the global smartphones becomes 4 TWh per year.

The capacity of production of a nuclear power plant varies much depending on factors such as the number of reactors. 2346 TWh of nuclear electricity production were registered in 2012. Fig. A.1 shows the year electricity production of the 437 operational nuclear power plants in the world. The cost of charging the smartphones corresponds to a relatively small power plant.

The average annual electricity consumption for a U.S. residential utility customer was 11280 kWh in 2011. Therefore, the energy spent by charging the smartphones is equivalent to the energy consumption of 354609 households (4 TWh / 11280 kWh) or the energy production of a small nuclear
Figure A.1: Energy production of the world’s nuclear power plants.

power plant.

---

Appendix B

Measurement setup

Fig. B.1 shows an overview of our measurement setup used for 3G and WiFi measurements. The setup is described separately for 3G and WiFi.

3G setup: The 3G measurements were performed on a power-efficient mobile broadband module (Ericsson F3307) provided by Ericsson AB (KRY 901214/01, marked as Development Kit in Figure B.1). Since this exposes interfaces to measure the power consumption of the 3G modem easily isolating it from the rest of system (e.g., CPU or screen), we employ this platform instead of the 3G module in a smartphone. The mobile broadband module provides 2G, 3G and GPS connectivity. It is designed to provide mobile broadband to consumer electronic devices such as tablets and laptops. The
module provides uplink and downlink speeds up to 5.76 and 7.2 Mbps respectively implementing the High Speed Packet Access (HSPA) standard.

The measurement setup depicted in Figure B.1 consists of the mobile broadband module placed on an Ericsson’s Developer Starter Kit. The kit provides network connectivity via USB to a test computer that runs a Ubuntu 10.10 with a firewall activated in order to allow only the desired network connections. The measurements were performed using a National Instruments myDAQ data acquisition device sampling the voltage drop over a shunt resistor of 0.1 (R1 in Fig. B.1) at 1 kHz. The power consumption is derived from the voltage and in order to avoid any anti-aliasing effects a low-pass filter of approximately 16 Hz (R2 = 10 k and C = 1 uF in the figure) was added.

Most measurements were performed using a Subscriber Identity Module (SIM) card from the network operator TeliaSonera providing full access to the available capacity of the Sweden 3G network. Unless specified, all the measurements were performed in the same location at university where the received signal strength did not vary significantly.

**WiFi setup:** The WiFi measurements were performed by removing the battery of the smartphone under test, and adding a low-side sensing circuit (R4 = 0.1Ω) with an isolating amplifier (maximum transmission error of 0.4%). Apart from V+ and ground pins, smartphone batteries often have a temperature and a communication line back to the phone. These pins are used to tell the phone the charge condition of the battery (smart battery interface) and when the charge cycle is complete. We added R3 (33 kΩ) in order to allow smartphones to switch on. The power consumption is sampled at 1 kHz using the myDAQ and the measurement values where compared against a digital multimeter (Agilent U1252B) for validation.

![Figure B.2: Aggregated power consumption for CPU and network transmission.](image-url)
Ideally, one would like to only measure the transmission energy (i.e., the energy spent by the peripheral hardware). We adopt a methodology to isolate the transmission energy from the rest of the system as in earlier works [99]. All smartphone components are switched off (e.g., screen, sensors or other radios). Since the operating system is preemptively multitasking, different processes are waking up and consuming resources such as CPU. An example of this effect is shown in Figure B.2, where it is hard to separate the energy consumed for the network transmission from the aggregated power consumption.

![Stabilised CPU power trace and a network transmission.](image)

Figure B.3: Stabilised CPU power trace and a network transmission.

In order to stabilise the power trace and isolate the transmissions from the rest of the system, we fix the CPU frequency and run a low-priority background process in a busy-loop. This stabilises the power trace. Figure B.3 shows the aggregated energy of a network transmission and the CPU with our methodology. The CPU creates an almost constant power consumption (0.37 W) that enables the isolation of the transmission energy.

The drawback of this technique is that we cannot distinguish between the CPU load created by the test and the background load. However, this should have little effect at the time of studying the peripheral hardware of the phone such as the wireless interfaces.
Appendix C

Baseline comparison with IM applications

A qualitative comparison with the popular IM applications is shown in this appendix. We are interested on knowing the energy consumption of the most basic 1-to-1 chatting functionality, thus our prototype is expected to have a lower energy consumption. The Short conversation is used as example and the test is repeated 5 times.

![Energy Consumption Graph](image-url)

Figure C.1: Energy consumption of the basic implementation compared to popular IM applications when sending the Short conversation.

Fig. C.1 shows the energy consumption of the different applications normalised to the base case (22.56 Joules). The energy cost of only sending the Short conversation increases 40% by sending it with GTalk instead of with the base implementation. Since applications are black boxes, no conclusions can be drawn from this result. However, it provides insight of how much room there is for implementing extra features before the base implemen-
APPENDIX C. BASELINE COMPARISON WITH IM APPLICATIONS

tation reacher the energy consumption of other applications. The energy consumption of Messenger seems unjustifiable based on the fact that it uses the same protocol.

![Figure C.2: Time spent in the different RRC states of the basic implementation compared to popular IM applications when sending the Short conversation.](image)

Fig. C.2 shows the time spent by the client in the different RRC states for a single run of the test. The least consuming applications make a more efficient use of the FACH channel by sending small packets. The main difference between the base implementation and GTalk is the amount of data sent, which makes the device to spend more time in the FACH state. Even though Viber switches the UE to DCH for a short period, it still consumes less than WhatsApp since the time spent in the PCH state is longer. The data sent by GTalk and WhatsApp made the UE to occasionally switch to DCH for short time (3 out of 5 tests for GTalk and 1 out of 5 for WhatsApp). However, efficiently using the DCH channel by sending data in bursts and keeping the UE in PCH for longer periods (e.g., GTalk and Viber) can be less consuming than continuous data transfer in FACH (e.g., WhatsApp). Data intensive IM applications that send big packets (e.g., Skype, Kik and Messenger) make the device spends most of the test time in the DCH state, consuming a large amount of energy. An energy-efficient IM application should avoid the DCH state when possible since the data to be sent is generally small.
Bibliography


<table>
<thead>
<tr>
<th>No</th>
<th>Name</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>Vojin Plavsic</td>
<td>Interleaved Processing of Non-Numerical Data Stored on a Cyclic Memory.</td>
</tr>
<tr>
<td>73</td>
<td>Ola Strömforss</td>
<td>A Structure Editor for Documents and Programs, 1986.</td>
</tr>
<tr>
<td>74</td>
<td>Christos Levopoulos</td>
<td>New Results about the Approximation Behavior of the Greedy Triangulation, 1986.</td>
</tr>
<tr>
<td>113</td>
<td>Ralph Rönquist</td>
<td>Network and Lattice Based Approaches to the Representation of Knowledge, 1987.</td>
</tr>
<tr>
<td>126</td>
<td>Dan Strömborg</td>
<td>Transfer and Distribution of Application Programs, 1987.</td>
</tr>
<tr>
<td>177</td>
<td>Peter Åberg</td>
<td>Design of a Multiple View Presentation and Interaction Manager, 1989.</td>
</tr>
<tr>
<td>181</td>
<td>Henrik Eriksson</td>
<td>A Study in Domain-Oriented Tool Support for Knowledge Acquisition, 1989.</td>
</tr>
</tbody>
</table>
No 380  Johan Ringström: Compiler Generation for Parallel Languages from Denotational Specifications, 1993.
No 558  Patrik Nordling: The Simulation of Rolling Bearing Dynamics on Parallel Computers, 1996.
No 563  Niclas Andersson: Compilation of Mathematical Models to Parallel Code, 1996.
Jörgen Lindström: Chefers användning av kommunikationsteknik, 1996.
Niclas Wahllöf: A Default Extension to Description Logics and its Applications, 1996.
FiF-a 4 Carita Åbom: Videomötesteknik i olika affärssituationer - möjligheter och hinder, 1997.
Fredrik Eklund: Declarative Error Diagnosis of GAPLog Programs, 1997.
Mats Gustafsson: Bringing Role-Based Access Control to Distributed Systems, 1997.
Bengt E W Andersson: Samverkande informationssystem mellan aktörer i offentliga åtaganden - En teori om aktörsressor i samverkan utbyte av information, 1998.


Charlotte Björkregen: Learning for the next project - Bearers and barriers in knowledge transfer within an organisation, 1999.


Svein Bergum: Managerial communication in telework, 2000.


Jean Paul Meynard: Control of industrial robots through high-level task programming, 2000.


Magnus Kald: The role of management control systems in strategic business units, 2000.


Fredrik Elg: Ett dynamiskt perspektiv på individuella skillnader i heuristisk kompetens, intelligens, mentala modeller, mål och konfidenstn i kontroll av mikrovärlden Moro, 2002.


Peter Bunus: Debugging and Structural Analysis of Declarative Equation-Based Languages, 2002.


Charlotte Stoltz: Calling for Call Centres - A Study of Call Centre Locations in a Swedish Rural Region, 2004.
Per-Ola Kristensson: Large Vocabulary Shorthand Writing on Stylus Keyboard, 2004.
Vaida Jakonienė: A Study in Integrating Multiple Biological Data Sources, 2005.
Abdil Rashid Mohamed: High-Level Techniques for Built-In Self-Test Resources Optimization, 2005.
Christina Keller: Virtual Learning Environments in higher education. A study of students’ acceptance of educational technology, 2005.
David Dinka: Role and Identity - Experience of technology in professional settings, 2005.
Andreas Hansson: Increasing the Storage Capacity of Recursive Auto-associative Memory by Segmenting Data, 2005.


Andreas Käll: Översättningar av en managementmodell - En studie av införandet av Balanced Scorecard i ett landsting, 2005.

He Tan: Aligning and Merging Biomedical Ontologies, 2006.


FiF-a 90


Jiri Trnka: Prerequisites for data sharing in emergency management, 2007.


Erik Kuiper: Mobility and Routing in a Delay-tolerant Network of Unmanned Aerial Vehicles, 2008.


Martin Karresand: Completing the Picture - Fragments and Back Again, 2008.


Mirko Thorstenstorp: Using Observers for Model Based Data Collection in Distributed Tactical Operations, 2008.


Mohammad Saifullah: Exploring Biologically Inspired Interactive Networks for Object Recognition, 2011.