



UNIVERSITY OF TRENTO - Italy

Department of Information Engineering and Computer Science

Master's Degree in  
Telecommunications Engineering

ANALYSIS AND EXPERIMENTAL COMPARISON OF THE ENERGY EFFICIENCY  
OF MOBILE CROWDSENSING DATA COLLECTION FRAMEWORKS

Supervisor  
Prof. Fabrizio Granelli

Candidate  
Mattia Tomasoni

External Supervisors  
Dr. Claudio Fiandrino  
Andrea Capponi

Academic year 2016/2017



# Summary

Nowadays, the growing population in urban area calls for a sustainable development of modern city in order to improve quality of life of the citizens. In such environment, sensing is essential to obtain informations useful to perform environment monitoring and provide services to the citizens such as urban safety and health care.

In recent years, Mobile Crowdsensing (MCS) become an interesting paradigm for urban sensing where citizens contribute data gathered by their smart devices. the research develops Different Data Collection Frameworks (DCFs) able to perform sensing and report the information to a central collector which provide shared resources to store and analyse the received data. As a consequence, investigate the energy efficiency of such DCF is crucial.

This works aims to develop a methodology to measure energy consumption and the amount of collected data of a data collection framework and evaluate the performance in a citywide scenario. As a first step, I develop an Android application able to perform data collection exploiting the underlying reporting mechanism of three representative DCF. Exploiting such application I perform measurements using a power monitor e Wireshark while sensing and reporting data in order to profile real energy consumption and amount of data collected for each of the considered DCF. In the second part, I feed CrowdSenSim simulator with aforementioned profiles for large-scale evaluation in different cities which differ for size and urban morphology. CrowdSenSim is a custom simulator specifically design at the University of Luxembourg to assess the performance of crowdsensing activities in large urban areas. The simulation results highlight the effectiveness of the frameworks allowing to evaluate their energy efficiency.

# Acknowledgements

I would like to thank my supervisors Prof. Fabrizio Granelli, Dr. Claudio Fiandrino and Andrea Capponi. Prof. Fabrizio Granelli is always available and gave me the chance to apply for the internship position at University of Luxembourg. Dr. Claudio Fiandrino and Andrea Capponi guided me during the entire research work and writing. I'm also really grateful to Piergiorgio Vitello for his precious help in performing simulations. Finally, I can not avoid to thank all my family for their unconditional support.

Their contribution allow me to reach the satisfying objective presented in this work.



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Motivation . . . . .	1
1.2	Contribution . . . . .	2
1.3	Thesis Organization . . . . .	3
<b>2</b>	<b>Background</b>	<b>5</b>
2.1	Mobile Crowdsensing . . . . .	5
2.2	Data Collection Frameworks under Analysis . . . . .	9
2.2.1	Deterministic Distributed Framework (DDF) . . . . .	10
2.2.2	Probabilistic Distributed Algorithm (PDA) . . . . .	10
2.2.3	Piggyback Crowdsensing (PCS) . . . . .	11
2.2.4	Performance Indicators for DCF Evaluation . . . . .	11
<b>3</b>	<b>Profiling Energy Consumption</b>	<b>13</b>
3.1	Data Reporting Mechanism . . . . .	13
3.2	The MCS Application . . . . .	14
3.2.1	REST: an Architectural Style for Web Services . . . . .	15
3.2.2	The Building Blocks for Programming a Crowdsensing Application in Android . . . . .	17
3.3	System Architecture . . . . .	25
3.4	Measurement Set-Up . . . . .	26
3.5	Experimental Results . . . . .	30
<b>4</b>	<b>Measurements-Aware Simulations: Results and Analysis of Data Collection Frameworks</b>	<b>35</b>
4.1	Large-scale Analysis: the Methodology . . . . .	35
4.1.1	CrowdSenSim . . . . .	36
4.1.2	CrowdSenSim for Data Collection Frameworks evaluation . . . . .	36
4.1.3	Simulation Setting . . . . .	39
4.2	Simulation Results . . . . .	42
4.2.1	Simulations Based on Datasheets Informations . . . . .	42

4.2.2 Simulations Based on Measured Energy Consumption and Data Generated . . . . .	46
<b>5 Conclusion and Future Works</b>	<b>51</b>
<b>Bibliography</b>	<b>53</b>
<b>Online resources</b>	<b>61</b>





# Chapter 1

## Introduction

### 1.1 Motivation

Cities have grown continuously in terms of area and population density during the last years and this figure is projected to increase in the next three decades. In addition, although cities occupied less than 3% of worldwide available surface they were already responsible for more than 70% of greenhouse gas emission and 60% of water use. [21]. Such situation implies the necessity to provide intelligent solutions in order to avoid an uncontrollable increasing of energy resources demand and permit a sustainable urban future.

The improvement of Information and Communications Technology (ICT) provide the instruments to design new services such as the Internet of Things (IoT) paradigm. It aims to include in the Internet environment a high variety of devices such as home appliances, cameras, smart devices and sensors in order to create new kind of interaction among "things" and humans. This kind of interaction defines an abstraction layer for many services based on distributed system and cloud computing such as storage (DaaS), infrastructure (IaaS), software (SaaS) and sensing (S<sup>2</sup>aaS) that nowadays are widespread diffused and available to the modern smart cities which aim to increase quality of life of citizens. Mobile devices (e.g., smart phones, tablets, wearables) play a very important role in smart cities solutions. They usually have a considerable computing power and many kind of embedded sensors such as accelerometer, gyroscope, magnetometer, microphone and camera but is also possible thank to the modern pairing technology like Bluetooth or WiFiDirect, to add external and more complex modules. Moreover they are also able to furnish GPS location. On the other hand, with the implementation of actual mobile infrastructures (3G/4G) and the constant increasing of their capabilities, the pervasive mobile computing is a growing trend where devices are constantly connected to the network and always available. Combining all this features, mobile devices can be considered as a part of a big and very powerful system able to

collect different kind of data which is useful to understand cities environment and provide services to citizens such as health care, urban safety, waste and emergency situation management and many other domains [72, 53, 34].

This system represents the base of Mobile Crowdsensing (MCS) paradigm. MCS follows a Sensing as a Service ( $S^2aaS$ ) business model, which makes data collected from sensors available to cloud users [7]. Consequently, companies and organizations have no longer the need to acquire an infrastructure to perform a sensing campaign, but they can exploit existing ones recruiting and compensating users for their involvement [23].

Due to the growing interest in data collection mobile applications, several works design different kind of Data Collection frameworks (DCFs) in order to feed the cloud collector. However, to be effective a MCS campaign needs to ensure large users participation. As a consequence, a proper users recruitment must take into account the energy efficiency exhibit by data collection in order to balance revenues and costs during the crowdsensing campaign.

## 1.2 Contribution

The main objectives of this work are to profile the energy consumption of different data collection frameworks and to include such profiles into CrowdSenSim simulator [15] for large scale evaluations in city-wide scenarios. Specifically designed at the University of Luxembourg CrowdSensim performs scalable MCS activities for a required duration (e.g., days) in complex environments. To this end, I have developed a novel mobile application capable to gather data from smart devices' sensors and GPS module and sending such data to the collector exploiting three different data reporting mechanism at the basis of the three families of data collection frameworks considered. I further used a power monitor and Wireshark while the application performs sensing and reporting activity and exploit several scripts in Python to obtain energy- and network-traces respectively. I chose smartphones based on Android framework, which is the most widespread operating system in mobile market (74% of market share - January 2018) [61]. Moreover, Google API implements various methods to manage embedded sensors and modules, such as accelerometer, gyroscope, light sensor and GPS. In order to store and analyze data I implement the collector using XAMPP, a framework that provides on top of it Apache webserver, PHP, PhpMyAdmin, and a MariaDb data base. For energy measurements the server application was installed on a laptop running Windows 10 operating system. The transmission of collected data was performed exploiting the Eduroam WiFi network. Eduroam is a secure, world-wide roaming access service developed for the international research and education community. Using Eduroam instead of an ad hoc wireless network provide a more realistic characterization of the transmission behaviour in real environment.

However, as MCS systems require large participation to be effective, performing experiments on real testbeds is often not feasible. To this end, simulations are a valid alternative. I extract theoretical value of energy consumption and amount of generated data from datasheets of commonly used hardware in mobile design and then perform measurement exploiting the power monitor and Wireshark. Hence, both the calculated values and the collected energy- and network-measurements were employed to feed CrowdSenSim simulator. The final outcomes are evaluated on the basis of three performance indicators (i.e., energy consumption, amount of data collected and fairness among the users) and shows how data related to the energy consumption values and traffic statistics obtained by measurements instead of theoretical models and datasheets are more descriptive of the real context and furnish to an organizer useful advises to evaluate which Data Collection Framework (DCF) fit better the requirements of a specific MCS campaign.

## 1.3 Thesis Organization

The document is subdivided into five chapters. This section provide for each of them a briefly explanation of the content.

- **Chapter 2:** this chapter provides an overview on the state of the art, in particular introduce the mobile crowdsensing paradigm, some practical applications, existing works for data collection and compare the characteristics of our Android application with some other already available. Next, presents the three DCF considered for the energy efficiency evaluation and the performance indicators considered.
- **Chapter 3:** this chapter described the implementation of the Android application, showing the system architecture and an overview about the measurement set up. The final part of this chapter present the results extracted by analysing the energy- and traffic-traces obtained from the power monitor and Wiresharks.
- **Chapter 4:** this chapter introduces the CrowdSenSim simulator used for large-scale evaluations and the simulation settings. Next, it presents both theoretical- and measurement-based results obtained using the energy consumption values and traffic statistics obtained by measurements instead of theoretical models and datasheets permit to have a precise view of how a particular activity impact in considered device.
- **Chapter 5:** this chapter concludes the work and presents future research directions on the topic.





# Chapter 2

## Background

This chapter starts introducing the mobile crowdsensing paradigm and presenting some practical applications for activity recognition, healthcare and environment monitoring. Provide an overview of the main works in literature and their Data Collection Frameworks (DCFs). Next, present major existing Android applications for crowdsensing data collection. The last section explain the characteristics of three representative DCFs of current state of the art took into account for the evaluation exposed in this work. Hence, the same section introduce also the performance indicators considered.

### 2.1 Mobile Crowdsensing

Mobile Crowdsensing (MCS) is a promising sensing paradigm which consists in gathering data from the mobile devices of a large multitude of users. The data collection can be participatory if relay on users active contribution or opportunistic if is not require any action to triggering the data collection [34]. Mobile devices report collected data to a central collector where it is stored at disposal of the organizer of a sensing campaign, such as a government agency, an academic institution or a business corporation. The collector is typically located in the cloud and provides shared services and shared resources to store, analyze and process the received data (see Figure 2.1). The purpose can be oriented to furnish back services to the crowd or organization such as municipalities or research institutes. The performance of MCS systems is strictly correlated to the crowd participation in the collection process. User recruitment is necessary to optimize resources, minimizing costs and maximizing return at the same time. In [17] Fiandrino et al. present an interesting approach to the problem considering not only the distance between users and sensing targets but also a sociability factor as estimation of their willingness to participate. In [33] is approached the problem of user recruitment exploiting the opportunistic networking paradigm. Depending on their mobility

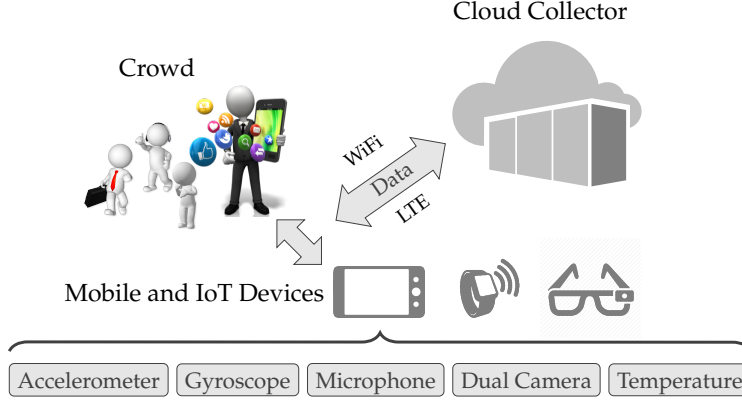


Figure 2.1. Cloud-based MCS system

patterns, the users involved in the data collection are able to act as a sensor or a relay for data delivery.

By exploiting smart device sensors instead of custom deployed sensor networks is considerably cheaper for the MCS organizer. However, in order to design a crowdsensing campaign is necessary also to consider how to reward users involved while sustain costs to perform data sensing and delivery, both in terms of energy spent and data subscription plan. Provide adequate incentives is a key point to guarantee a large participation. User rewarding is managed by different strategies e.g., monetary-based reward models [71] or a socially-aware system in order to exploit user's social trust [18].

Since in MCS systems there is no a direct control of users behaviour, they may exhibit several level of personal effort and as a consequence collects data of different quality. Indeed, data may be affected by the difficulty of sensing tasks, the characteristics of mobile sensors, the description of the MCS activity by the organizer and the user's abilities to collect data or its willingness [56]. In order to avoid this issues, it is possible to exploit different strategies such as directly intervene on the users motivation to collect reliable data using adequate incentive mechanism [49], validate the data considering the reputation of the user [73] or perform data evaluation during the analysis phase [28, 27].

Many practical applications take advantage of MCS paradigm to furnish different kind of services. Activity recognition-based applications are categorize in three different groups considering who is the beneficiary. In particular, such applications can provide useful information to end users (e.g., fitness tracking, health monitoring, fall detection) developers or a third parties (e.g., targeted advertising, Corporate management) or crowd/groups (e.g., activity-based social networks, place & event detection) [40, 8, 10, 25, 14, 43]. Healthcare applications

allow medical researchers and doctors to constantly monitor users condition in order to highlight symptoms not visible during medical examinations or detect deterioration of a patient's current physical status [52, 41, 4]. MCS applications also provide to organizations real datasets on which extract information useful to managing traffic jam and urban parking [48, 13], road condition [11], noise and air pollution [54, 60, 42] and make possible to create a monitoring system for the surrounding environment. Creekwatch [35] is an application for smartphones developed by the IBM Almaden research center. It allows the monitoring of the amount of water in the river bed, the amount of trash in the river bank, the flow rate, and let users to take a picture of the waterway. Garbage Watch [55] employs citizens to monitor the content of recycling bins with the objective of improving the recycling program. Such applications can exploit different kind of data reporting, the implementation choice is driven by the application field. Indeed, the reporting phase can be delayed-tolerant if is not necessary to deliver data as soon as collected. On the contrary, data is delivered in a continuous fashion for applications which required to monitor phenomena in real-time or also following a probabilistic approach which is an intermediate solution where data is delivered as soon as possible but not strictly after sensing.

In order to perform data collection, crowdsensing applications exploit different Data Collection Frameworks (DCFs). Several works propose different DCFs in order to enhance performances of MCS activities satisfying requirements such as energy efficiency and amount of collected data. Wu et al. [68] investigate the trade-off between the amount of acquired data and the associated energy consumption. The authors present and analyze both off-line and on-line settings for task allocation. In off-line case, the entire task information is known a-priori and does not change over time, while in the on-line scenario tasks are dynamically allocated in real-time without any information in advance. The authors first provide an optimal algorithm for the off-line setting. Then, they investigate the on-line setting where requests arrive dynamically without prior information in a first-in-first-out (FIFO) manner or with an arbitrary deadline (AD). Wang et al. [64] investigate the problem of scheduling multiple sensing tasks with the objective of ensuring the quality of sensed data while minimizing the energy consumption. Starting from basic cases in which sensing process requires data from only one sensor, the authors define the Minimum Energy Single-sensor task Scheduling (MESS) problem and design a polynomial-time optimal algorithm. Then, they consider a generic case in which sensing tasks require data from multiple sensors to be accomplished. To solve the problem of Minimum Energy Multi-sensor task Scheduling (MEMS), the authors propose an Integer Linear Programming (ILP) formulation as well as two effective polynomial-time heuristic algorithms. In [74], the authors propose a fair energy-efficient allocation framework whose objective is to minimize the maximum aggregated sensing time. The problem is NP-Hard

also when the information on the tasks such as arrival and duration is known prior to the allocation. The authors first investigate the off-line allocation model and propose an efficient polynomial-time approximation algorithm with a factor of  $2 - 1/m$ , where  $m$  is the number of mobile devices joining the system. Then, focusing on the on-line allocation model, they design a greedy algorithm which achieves a ratio of at most  $m$ . Han et al. [24] propose an on-line learning algorithm, where a central authority assigns tasks aiming at rewarding participants with a limited amount of budget. It supposes a fixed minimum number of users who actively join the sensing process, while the quality of collected data may vary. Liu et al. [38] propose a method to efficiently select users for participatory crowdsensing. Contributors are dynamically chosen considering their willingness to acquire data and their potential, which is calculated considering the remaining battery in their smartphones. The distribution of tasks aims to minimize the probability that an individual does not accomplish the assigned task. The CARDAP [30] DCF exploits a fog computing platform to enable efficient data analytics performed in a distributed fashion. The fog allows CARDAP to extend and augment functionality of a previously proposed general-purpose framework called CAROMM [59]. Similarly to CARDAP, the framework proposed in [16] exploits the fog to perform user recruitment based on multiple criteria, including distance of the participants from the location of the sensing task, their remaining battery charge and the user sociability defined in terms of the amount of time and data users exchange through social media. Wang et al. [65] present an algorithm to report information in an energy-efficient way. It classifies users into two groups. In the first category, the target is to minimize the energy consumption while reporting data and the individuals pay for the data they utilize to the operators. In the second group, users aim to minimize the cost of data reporting using communication technologies such as WiFi or Bluetooth, which are free-of-charge.

This work aims to compare the performances of three representative DCFs evaluating their energy efficiency, amount of data collected and fairness among the users. To the best of our knowledge, there are no existing studies performing such analysis. The closest to our work is [50], where Peng et al. investigate the trade-off between energy efficiency and fairness between participants. They propose an allocation framework which aims to minimize the maximum aggregate sensing time and for the experimental setup they exploit the Power Monitor to assess the power consumption of a smartphone under different configurations.

In order to perform the large-scale simulations I developed a novel RESTful Android application able to collect data performing each of the three reporting mechanism (i.e., delayed, continuous, probabilistic) underlying the considered DCFs and introduced in Section 3.1. REST guidelines provide an architectural style for developing web services and its philosophy is explained more in detail in Subsection 3.2.1. Other works in literature require to design similar applications

for crowdsensing data collection but none of them provide the same features. SystemSens [12] is a tool developed to capturing the usage context during data collection. It is composed by an Android logging client that collects device's usage parameters and a visualization web service. The tool aims to provide context information useful to better understand the external anomalies affecting the data collected. Moreover, it can furnish debugging information for unexpected battery consumption. SystemSens fall in the delayed-tolerant application performing data reporting when the considered device is being charged. APISENSE [22] is a popular cloud-based platform that enables researchers to deploy crowdsensing applications by providing resources to store and process data acquired from a crowd. It presents a modular service-oriented architecture on the server-side infrastructure that allows researchers to customize and describe requirements of experiments through a scripting language. In addition, it makes available to the users other services (e.g., data visualization) and a mobile application, allowing them to download the tasks, to execute them in a dedicated sandbox and to automatically upload data on the server. Mulero et al. [46] present an infrastructure for smart cities that combines IoT and Linked Open Data paradigms to provide a scalable and responsive system able to provide services. They exploit a REST API that receives and manages a large amount of data. CRATER [31] is a crowdsensing platform to estimate road conditions. It provides RESTful APIs to access data and visualize maps in the related application.

The next Section present the data collection frameworks considered in our study and introduce the performance indicators used to compare them.

## 2.2 Data Collection Frameworks under Analysis

MCS systems aim to minimize both sensing and reporting energy consumption. The most relevant part of the overall amount of energy is consumed during the reporting phase defined by the Data Collection Framework (DCF) used to acquire data. The DCFs can be general purpose or application-specific. The first typology has the capability to work with different application at the same time while the second one is designed for a specific purpose.

In this work we considered three general-purpose opportunistic DCFs representing three different families of data acquisition methodologies characterized by properties and features highlighted in the following paragraphs. Other existing DCFs in the literature exhibit minor variations with respect to the chosen ones. The main differences concern the data reporting phase and can be classified as intermediate solutions of these three main families.

### 2.2.1 Deterministic Distributed Framework (DDF)

DDF is a general-purpose framework for energy-efficient data collection in opportunistic cloud-based MCS systems proposed by Capponi et al. [6]. It aims at maximizing the utility of the cloud collector in receiving data from certain sensors in a specific region of interest, while minimizing at the same time the energy costs users sustain to sense and deliver information. The central collector periodically sends to mobile devices beacons to advertise the utility in receiving data from specific sensors in a certain area. Then, the sampling decisions are taken in a distributed fashion at each mobile device locally. Sensing and reporting decisions are driven by environmental context, an estimation of the potential utility and cost of doing sensing and reporting, the level of battery and the amount of data already contributed, and several other parameters. Therefore, the mechanism considers the previous history of the users to determine whether to perform next sensing and reporting operations. This enables fairness among users because prevents data collection from users whose level of battery is too low or that have already contributed considerably in the past.

The applicability of DDF spans across multiple scenarios of interests for smart cities, such as real-time monitoring of the environment or intelligent transportation systems. Such application scenarios require continuous data reporting for an up-to-date analysis of the status of the phenomena observed.

### 2.2.2 Probabilistic Distributed Algorithm (PDA)

Montori et al. [45] propose a distributed algorithm based on probabilistic design to acquire data in an opportunistic scenario. The algorithm is based on a limited feedback from the central collector and does not require users completing specific tasks, hence it is in line with the spirit of generic-purpose DCF. The objective of this algorithm is to regulate the amount of data contributed from users in a certain region of interest to avoid data redundancy and energy waste. Additionally, the algorithm aims at providing fairness to the users. Assuming that it is impossible to compute the number of participants in a region of interest because their position is not tracked, the coordinator estimates the required number of participants by computing the number of observations already acquired. The central platform is responsible to set a total per-zone number of observations required to reach a certain level of accuracy in observing a given phenomenon. To reach this goal, the mobile devices decide independently from the central authority whether to perform sampling and reporting. The framework is memoryless because users contribute data independently from the level of previous participation. The range of scenarios where PDA is applicable falls into the same category of DDF.

### 2.2.3 Piggyback Crowdsensing (PCS)

PCS [36] is a DCF that aims at reducing at the minimum any energy cost to promote user participation. The collector does not provide any form of coordination to trigger sensing decisions. Data reporting occurs during the so-called smartphones' opportunities, i.e., sensed data is piggybacked during phone calls or when connected-applications exchange data with remote servers. During these opportunities, the overhead of performing data reporting is low because mobile devices do not have to wake up the radio interface to transmit the collected data on purpose. All the aforementioned features makes PCS suited for delay tolerant MCS tasks that do not need data to be sent to the central collector in real time. For instance, PCS could be exploited for mapping non-real time phenomena like air quality or noise monitoring, requiring only time and place labels or check-ins in mobile social networks.

### 2.2.4 Performance Indicators for DCF Evaluation

In order to evaluate The Data Collection Frameworks (DCFs), we introduce in this Section the performance indicators considered.

#### Energy Efficiency

A very important aspect necessary to take into account is the energy efficiency. While devices continuously increase their hardware performances and as a consequence energy demand, the battery capacity does not follow the same trend discouraging users participation. The highest amount of energy is usually drain by the display and network connectivity [32], but also application development can introduce energy inefficiency problems. GreenDroid [39] is a tool aims to help developers in finding such inefficiencies in their application. It was developed after a huge analysis of more than 400 Android applications that reveal some common causes such as missing deactivation of sensors or wake locks. In order to foster the users to contribute in MCS campaigns is necessary to avoid energy waste while performing sensing and reporting phases. *EMC*<sup>3</sup> [70] is a framework for energy consumption optimization that aims to minimize task assignment avoiding redundancy. Xiong et al. [69] present a framework able to avoid energy waste sending task assignment and sensed data during phone calls.

#### Amount of Data Collected

The main objective of any DCF is to gather a sufficient amount of information to capture and or monitor phenomena. Hence, the quantification of the amount of data that the crowd can harvest is important to assess. Other performance

indicators related to Quality of Information (QoI) and accuracy of data are equally important, but in this work we focus solely on the amount of contributed data as the main objective is to compare the DCFs.

### **Fairness**

Fairness aims to evaluate the methodology in which a particular DCF provide reward for users' effort. Although a fair rewarding mechanism allow to have a proper users participation, there is only few works in literature considering fairness as a useful performance indicator. As examples, Huang et al. [26] present a task allocation mechanism based on the max-min fairness. In [47] Ni et al. present a study where they consider fairness in costs and efforts sustained by users in vehicular crowdsensing.



## Chapter 3

# Profiling Energy Consumption

In order to evaluate the Data Collection Frameworks (DCF) considered, I exploit a power monitor and Wireshark to obtain energy- and network-traces while the Android application I specifically developed performs sensing and reporting activities. This chapter describes how to obtain the aforementioned measurements. In particular, introduce the three Data Reporting Mechanism (DRM) defined to characterize different families of data collection frameworks, explain the design process necessary to develop the Android application and the architecture of the data collection system. The last part expose the measurement set-up and results.

### 3.1 Data Reporting Mechanism

A data reporting mechanism is the most important component of a data collection framework that defines the methodology to perform delivery of sensed information to the cloud collector. A DCF consists of multiple components in addition to the DRM, such as mechanism to inform the users about the urgency of sensing additional data. Every DCF implements a DRM: the ones presented in the following paragraphs are three methods that represent the most classical techniques for data delivery.

#### **Continuous-DRM (CON)**

In continuous reporting, data is delivered in a continuous fashion as soon as it is sensed. This approach is needed for real-time applications where users need to feed the collector with data constantly over time. However, such continuous stream of data incurs in the highest energy cost from the user point of view as the mobile devices need to maintain the connection active during the entire sensing operation period. The CON method is implemented by Deterministic Distributed Framework (DDF - see Subsection 2.2.1).

**Delayed-DRM (DEL)**

Delayed reporting starts to send data after the sensing activity has ended. Hence, this approach decouples sensing and reporting operations and is more conservative than CON from an energy-consumption perspective. Indeed, the network interface is not active during sensing period and the transmission of aggregated data minimize the networks overhead and errors retransmissions. DEL is useful for delay-tolerant applications where the collector does not need to harvest data in real-time, it is implemented by Piggyback Crowdsensing (PCS - see Subsection 2.2.3) framework when only 1 smartphone opportunity occurs.

**Probabilistic-DRM (PRO)**

This approach considers a probabilistic reporting. It is an intermediate solution between the two previous approaches. During each timeslot, the algorithm randomly generates a probability  $p$  that drives the decision of delivering the sensed data. The parameter  $p$  is checked against  $\delta$ , a feedback provided by the collector in order to indicate the necessity to receive new data by specific users. Its energy consumption depends on how many times a transmission occurs. PRO is implemented by Probabilistic Distributed Algorithm (PDA - see Subsection 2.2.2) and by PCS when the number of smartphone opportunities is higher than 1.

## 3.2 The MCS Application

The mobile application has developed exploiting the Android framework and as shown in Subsection 3.2.1 follows the Representational State Transfer (REST) guidelines. The GUI (see Figure 3.1) presents two different buttons to start/stop the foreground service used to manage all the fundamental operations such as gathering and saving data, ask to the server necessary information to perform the data collection (e.g, sensing time, server-side database state) and manage the data transmission to the collector. Moreover, the graphical interface shows additional information about the current operation such as notification, number of records present in the database and a progress bar to show the percentage of transmitted data. Finally, the application provides also two type of log activities. The first sends runtime coarse information directly to the server in order to follow the overall state of the data campaign and the second one writes detailed information directly on the private memory of the application. When service is stop, the user is able to select such detailed log directly from the application GUI and sharing it if necessary.

The main objective considered during the development process was to allow the user to perform MCS data collection exploiting any of the data reporting

mechanism presented in previous Section 3.1. Moreover, I can exploit this feature to measure the energy consumption and the network traffic generated for each DRM. Next Subsection introduce the REST guidelines highlighting the main advantages derived from a RESTful implementation of web applications.

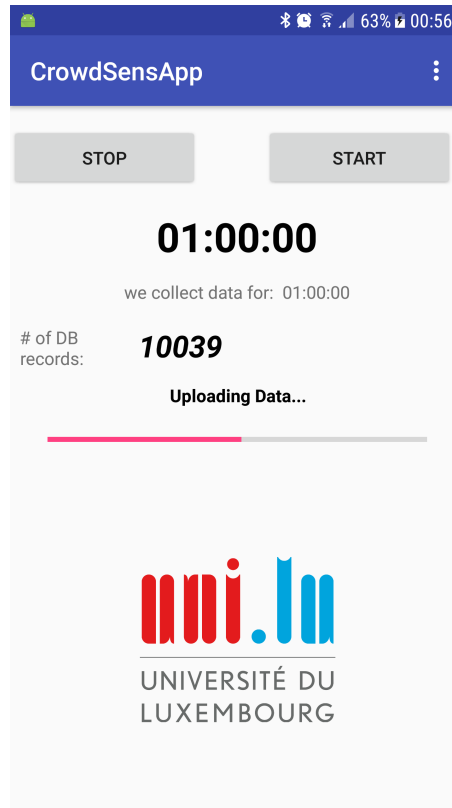


Figure 3.1. Screen-shots of the MCS application developed during my internship at the University of Luxembourg

### 3.2.1 REST: an Architectural Style for Web Services

Similarly to previous works in the area, I followed the REST architectural style to develop the application exploiting basic HTTP methods (e.g., GET and POST to exchange messages and data with the cloud) and HTTP response. Furthermore, the application has a client-server architecture and stateless communications because each message is self-descriptive through HTTP headers. In other words, to minimize the server workload each request has enough information for the server to process that message.

In general, a RESTful application complies the following guidelines [57] to achieve properties such as scalability, portability and high performances.

## Uniform Interface

The first guideline defines the interface between clients and servers. Uniform interface decouple the architecture in order to enabling each part to evolve separately and adding new ones if necessary. The principles of the uniform interface guideline are:

- *Resource-based*: resources are the fundamental elements of a RESTful web service (e.g, web addresses, files, services). They are uniquely identified using an Uniform Resource Identifier (URI). As an object in the object-oriented programming, a resource represent any entity in which is possible to perform operations. Such entities are conceptually separated from the representations returned to the client. As an example, The server does not send its database but an element (e.g, Json string) representing a database record.
- *Manipulation of Resources through Representations*: clients interact with resources using their representations. When a client holds a representation it has enough information to modify or delete the resource on the server.
- *Self-Descriptive Messages*: each message have all the information necessary to describe how to process itself.
- *HATEOAS*: Hypermedia as the Engine of Application State, this principle permit a client-server interaction using uniquely the hypermedia returned by the server application. Hence, a REST client does not need to know any information in advance.

## Stateless Communication

In stateless communication each client request incorporate all the necessary information (e.g., parameters, context, data) in order to allow server-application to perform operations and return back the result. In other word, each request is independent from each other. Stateless communication aims to avoid server overload releasing computational capabilities as soon as possible.

## Caching

In a RESTful architecture, resources are explicitly state as cacheable or not to prevent clients to reuse old or inappropriate data. Caching are able to reduce the client-server interaction improving scalability and performance.

## Client-Server

Client-server architecture provides a total separation of concerns between clients and servers. In this way, the portability of client code improve and server can be simpler and more scalable. Servers and clients may also be replace and/or improve independently, as long as the interface is not altered.

### 3.2.2 The Building Blocks for Programming a Crowdsensing Application in Android

This Subsection shows the main characteristics of the Android framework approached during the development of the crowdsensing application. Before to explain the system architecture in Section 3.3 is important to define the classes involved and their main setting parameters. Sensors management, location management, services and resource management are the basic bricks necessary to build an application able to collect different kind of data.

## Sensors Management

As a first step, I approached the management of the embedded sensors. The very first runnable code just activate the sensors available on the smartphone and a sensor listener in order to receive the sensed values and show them on the screen. Indeed, Android sensor framework allows to acquire raw sensor data using the following classes and interfaces:

- *SensorManager*: provides various methods for accessing and listing sensors, registering and unregistering sensor event listeners, and acquiring information. Moreover, also provides several sensor constants that are used to report sensor accuracy, set data acquisition rates, and calibrate sensors.
- *Sensor*: this class is used to create an instance of a specific sensor and provides various methods that permit to determine the sensor capabilities such as maximum range, its resolution, and its power requirements.
- *SensorEvent*: the system uses *SensorEvent* class to create a sensor event object, which provides information about sensed data. In particular, when a sensor event occur, it is possible to extract by the object returned: the raw sensor data, the type of sensor that generated the event, the accuracy of the data, and the timestamp related to the event.
- *SensorEventListener*: it is an interface with callback methods. A callback method receive notifications (i.e., sensor events) and will executed when a sensor value or accuracy change in order to manage the new data.

When sensor manager starts the sensor listener, is necessary to pass to the register method the parameter *samplingPeriodUs* that defines the maximum sample delay.

```
mSensorManager.registerListener(SensorEventListener listener, Sensor sensor,  
    int samplingPeriodUs);
```

Android provides 4 predefined sampling rate, but also allows (since API level 11) a custom value, as shown below:

- *SENSOR\_DELAY\_FASTEST*: get sensor data as fast as possible
- *SENSOR\_DELAY\_GAME*: rate suitable for games
- *SENSOR\_DELAY\_NORMAL*: rate (default) suitable for screen orientation changes
- *SENSOR\_DELAY\_UI*: rate suitable for the user interface
- *Custom value*: numeric value in millisecond

After several test I decide to exploit for data collection purpose the *SENSOR\_DELAY\_NORMAL* value in order to have a consistent data-set and at the same time, limit the energy consumption and the network traffic generated to report data to the cloud collector.

## Working with FIFO queue

Sensors available on smart devices, can exploit an hardware queue where new events are store until they have the possibility to be delivered. In order to get information about queue availability and length, the class *Sensor* implement the following two methods:

- *Sensor.getFifoMaxEventCount()*: returns the maximum length of the queue. If it is zero, any events can be batched because this functionality is not supported.
- *Sensor.getFifoReservedEventCount()*: returns the maximum reserved length when FIFO queue is shared with other sensors.

Moreover, during the listener registration provided by the sensor manager, another input parameter can be set in addition to the sample frequency mentioned above. The *maxReportLatencyUs* parameter.

```
SensorManager.registerListener(SensorEventListener listener, Sensor sensor,  
    int samplingPeriodUs,  
    int maxReportLatencyUs );
```

Respect to the previous registration where only the sample frequency is set, in this case events are allow to stay temporarily in the hardware FIFO queue before being delivered. In other word, if *maxReportLatencyUs* is not set, the system try to deliver new data as soon as it is sensed.

However, in the next Subsection is explained why the specified sampling frequency and latency delay are only a suggested value. The Android system and other applications can alter these values. For this reason they could be set at least equal to the minimum required value necessary to the application to work fine.

## Sensor Stack

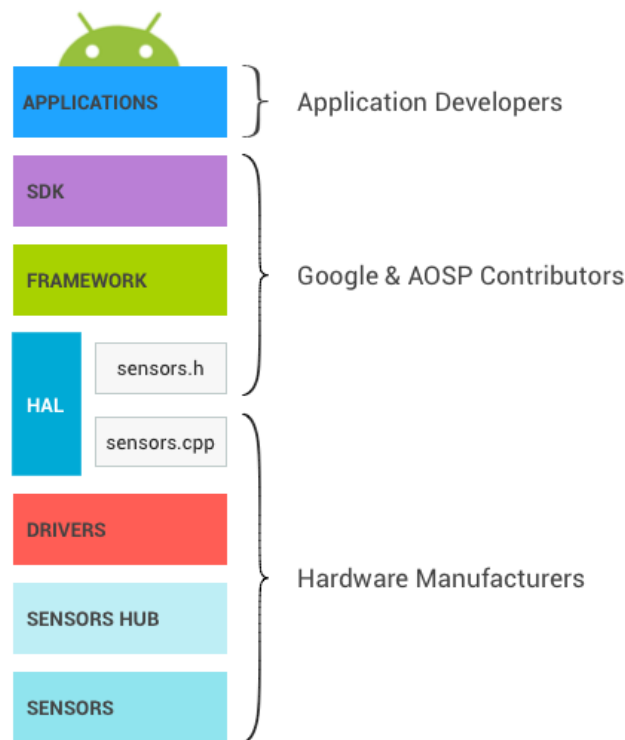


Figure 3.2. Layers of the Android sensor stack and their respective owners

The Android sensor stack (Figure 3.2)[20] shows how control flows from the Application layer to the sensor and data flows in opposite direction. Every block communicates only with neighbour blocks.

SDK layer (API) allows the Application layer to access the device sensors. The Framework provides links to all the applications to the HAL (Hardware Abstraction Layer). Such multiplexing is necessary to guarantee multiple access otherwise only one application can use a sensor at a given time. The HAL provides

APIs necessary to interface the Framework layer and hardware's driver. Sensor HUB is not always part of the stack and its architecture and protocol used to communicate with sensors is not specified by Android framework. It is useful to perform basic computation when the CPU is in suspended mode. It is also the layer where to perform sensor queuing.

In order to provide a fair access to the hardware resources, is necessary to take into account requirements from each application. Indeed, the framework aims to respect application expectations and working modality allowing only the configuration of sampling frequency and reporting latency. Nevertheless their are not guaranteed at the Application level. Considering a specific sensor:

- The sampling frequency will be the highest value among all the application requests. If just an application sets the sampling frequency using the *SENSOR\_DELAY\_FASTEST*, all application will receive new sensor value as fast as possible.
- The maximum reporting latency will be the lower among all the application requests. If just an application asks a latency equals to 0, all applications will receive event from this sensor in continuous mode (i.e., as soon as they are available).

Moreover, is not possible to send commands from the highest level of the stack to the sensors layer in order to avoid the possibility that an application modify the behaviour of the sensor and compromise the functionalities of other applications.

### **Sensor Type: wake-up or not wake-up**

Due to the fact that Android assumes is not necessary for most of the sensors to pass current values to Application level if users do not have any interaction with the device, sensors in smart devices are subdivide in two categories:

- Wake-up sensors.
- Non wake-up sensors.

Wake-up sensors, send an interrupt to the CPU every time an event occurs. On the contrary, non wake-up type can be delivered only if CPU is not in suspended mode. Both such kind of sensors can be provided with an hardware FIFO queue that can store a certain amount of events, so different scenario are possible as illustrated in next paragraphs.



### **Wake-up sensor and queue available**

This is the optimal case, it is possible to set a maximum delay for every events to let them wait in the queue before wake-up the CPU. In this way, system has the possibility to collect more information and pass to the application at once. CPU receives an interrupt only when maximum delay elapse or if queue length is full.

### **Wake-up sensor and queue not available**

This situation present the same behaviour of having a wake-up sensor with maximum delay set to zero. All the events are reported to the application as soon as they happen.

### **Not wake-up sensor and queue available**

In this case when CPU enters in suspended mode, events are buffered in the queue until some interaction turn it back to active operation mode. If the number of events stored exceed the length of the queue, older events are overwrite.

### **Not wake-up sensor and queue not available**

This represents the worst case. When CPU enters in suspended mode there is no possibility to receive information from sensor.

## **Location Request**

User location is a very important information to geo-reference sensed data. In Android, GPS data delivery can be conditioned not only by the application of interest but also by any other process that required such information. Hence, similarly to the management of on-board sensors data, all location requests are considered hints, and a specific application may receives locations that are more/less accurate, and more/less frequently than requested. The two main methods used to set a location request are:

- *setInterval(long)*: sets the desired time interval between each update.
- *setPriority(int)*: sets the priority of the request. This method specify the weight to be assigned to accuracy and energy spent.

The Android framework accepts any desired update interval simply passing the value as input to the *setInterval(long)*. In order to develop an application able to collect geo-referenced data I performed different on-field tests to define the better trade-off among a dense information about user's location and energy drained

from the battery. In such tests the location update interval was set to different values starting from 20 milliseconds to 200 seconds and using the application continuously until the end of the battery capacity on my Samsung Galaxy S6 which mount a 2550 mAh Li-Ion battery. As a result, I observed a battery lifetime varying from a couple of hours to 8-9 hours.

The battery lifetime can be affected also by the priority parameter. In particular, is possible to set four types of priority [19].

- *PRIORITY\_HIGH\_ACCURACY*: such kind of priority is used to request the most accurate locations available.
- *PRIORITY\_BALANCED\_POWER\_ACCURACY*: used to request "block" level accuracy. Location requests exploiting a block level accuracy return coordinate at most 100 meter far from the real position. Using a coarse accuracy such as this often consumes less power.
- *PRIORITY\_LOW\_POWER*: used to request "city" level accuracy. Location requests exploiting a city level accuracy return coordinate in the range of 10 km far from the real position. Setting such kind of accuracy allows to consume less power.
- *PRIORITY\_NO\_POWER*: used to request the best accuracy possible with zero additional power consumption. No locations will be returned unless a different process has requested location updates in which case this request will act as a passive listener to those locations.

```
LocationRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);
```

The code snippet shows an example where is set a high accuracy for the location calling the *setPriority(int)* method.

## Resource Management Policy

In [2] Google explains the lifecycle of processes and applications. In Android, every application runs in its own Linux process. This process is created when the application starts and will remain running until it is no longer needed and the system reclaim its memory for use by other applications. Applications can not control directly their process lifetime that it is determined by the system observing which parts of the application are running, how important they are for user experience and much memory is available. To decide which processes to kill, the Android system weighs their relative importance to the user. For example, a process without any activity visible at screen is a better candidate compared to a process hosting visible activities. When low on memory, Android places each

process into an "importance hierarchy" based on the components running in them and the state of those components.

### **Foreground Process**

A process is considered in foreground if it is required for what the user is currently doing. We can consider as foreground process an activity at the top of the screen, a broadcastreceiver process currently running or a service currently executing code in callback methods.

### **Visible Process**

A visible process is doing work that user is aware of. Killing it can have negative impact to user experience. A visible process can be an activity visible on the screen but not in the foreground, a process hosting a service using by the system for particular feature necessary to the user, such as live wallpaper or input method service. A visible process is also a service running as a foreground service. This is possible calling the *startForeground()* method which asks to the system to treat the service as something the user needs.

### **Service Process**

This process holds a normal service started without a call to the *startForeground()* method. A Service usually performs operations important for the user such as background data download/upload or every heavy operation that the main thread can not perform without compromise the user experience.

### **Cached Process**

A cached process is not currently needed. It can be killed by the system at any time if a memory lack occur. In optimal situation, a system have multiple cached process and when necessary starts to kill them from the oldest one.

### **How to Set a Process in a Crowdsensing Application**

During the development of the MCS application, the service responsible to gather, save and start the transmission thread when Internet connection is available is set as a visible process calling the *startForeground()* method as mentioned above. In such a way users are able to start the service and continue to use their smartphone and other applications avoiding the possibility in which the system easily stop the data collection.

## Connection Management

The Android framework provides also specific classes to manage Internet connections. In particular, *HttpURLConnection* class extends *URLConnection* adding specific features for http connections. Http is the transfer protocol commonly used to design RESTful application.

The use of this class needs to follow a specific pattern:

- First, definition of the request. The most important property is its URI.
- The next step consists in obtaining a new object casting the result of *URL.openConnection()*.
- Upload of the body message (if necessary) by calling the methods *HttpURLConnection.setDoOutput(true)* and *HttpURLConnection.getOutputStream()*.
- Read the response provide by the server. The response body may be read from the stream returned by *HttpURLConnection.getInputStream()*.
- Disconnect the http section by calling the *HttpURLConnection.disconnect()* method.

The *HttpURLConnection* allow to setting up an http connection defining the connection timeouts, input and output streaming and the type of the request which identify once among the typical CRUD operations (Create Read Update Delete). The connection thread of the MCS application developed used only GET and POST http methods.

- *HttpURLConnection.setRequestMethod("GET")*: GET provides information about the state of the server database or necessary to define the campaign (e.g, sensing duration).
- *HttpURLConnection.setRequestMethod("POST")*: POST is the method employed to create new databases.

The messages send through such methods is encoding using the *Json* format and contain all the necessary information to be process by the server such as the database ID. Both part are completely independent while the interaction between them is performed exploiting only commonly supported protocols and standards such as http and Json. These implementation choices guarantee the "server-client architecture" and "stateless communication" REST guidelines.

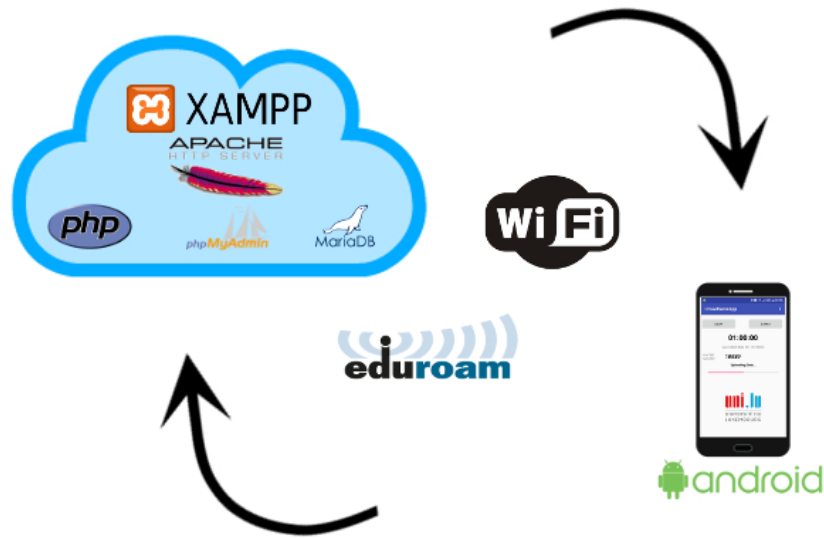


Figure 3.3. System architecture

### 3.3 System Architecture

The mobile application I developed can run over any Android-based smartphone, it is fit for setting which connectivity is possible to use for data reporting among WiFi, cellular network or both. Figure 3.3 illustrates the architecture with all its components. The figure also highlights the WiFi network EDUROAM that provides interconnection between the smartphone and collector during the energy measurements. Java is the programming language employed for the implementation, while PHP is the server-side scripting language used for the web services development. The minimum supported version is Android Marshmallow 6.0 (API level 23), but the application is already compatible with Android Oreo 8.0 (API level 26). As is possible to see in Table 3.1, such implementation decision is supported by the overall diffusion of last three version of Android (Marshmallow, Nougat, Oreo) which is above 50% [1]. The server side, i.e., the cloud collector is a laptop used to perform data processing and storage. In particular the system exploits XAMPP (v7.1.8 - 32bit), that provides in a unique distribution Apache web server and phpMyAdmin to manage the database. The most recent version of XAMPP features a database based on MariaDB.

The unique features of each Data Collection Framework (DCF) impact the DRM implementation design. For example, delayed-DRM requires a database to locally store the sensed data. continuous- and probabilistic-DRM are less strict and will work correctly if the sensed data is simply stored on a local buffer. A database ensures higher reliability as it adds a further level of protection at application

VERSION	CODENAME	API	DISTRIBUTION
2.3.3 – 2.3.7	Gingerbread	10	0.4%
4.0.3 – 4.0.4	Ice Cream sandwich	15	0.5%
4.1 – 4.3	Jelly Bean	16 – 18	5.6%
4.4	Kitkat	19	12.8%
5.0 – 5.1	Lollipop	21 – 22	25.1%
6.0	Marshmallow	23	28.6%
7.0 – 7.1	Nougat	24 – 25	26.3%
8.0 – 8.1	Oreo	26	0.7%

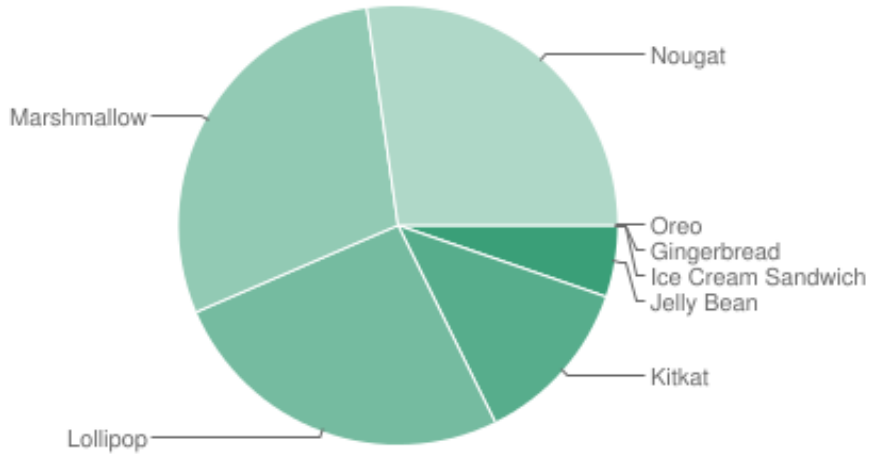


Table 3.1. Diffusion of Android version. Data collected by Google on January 8, 2018

layer on top of communication retransmissions schemes at lower layers of the protocol stack. Moreover, storing data in local database avoid the possibility to lose all the information harvested if the application drastically stops to work for any reason (e.g., smartphone reboot, battery death). Indeed, a database permanently stores the collected data allowing the MCS application to control the state on the server-side and eventually restart the synchronization. On the other hand, due to the more complex operations, working with a database require a little bit more computational power and as a consequence increase the energy cost.

### 3.4 Measurement Set-Up

This Section presents the measurement set-up prepared to obtain energy- and network-traces. Energy profiles of the data reporting mechanism are obtained

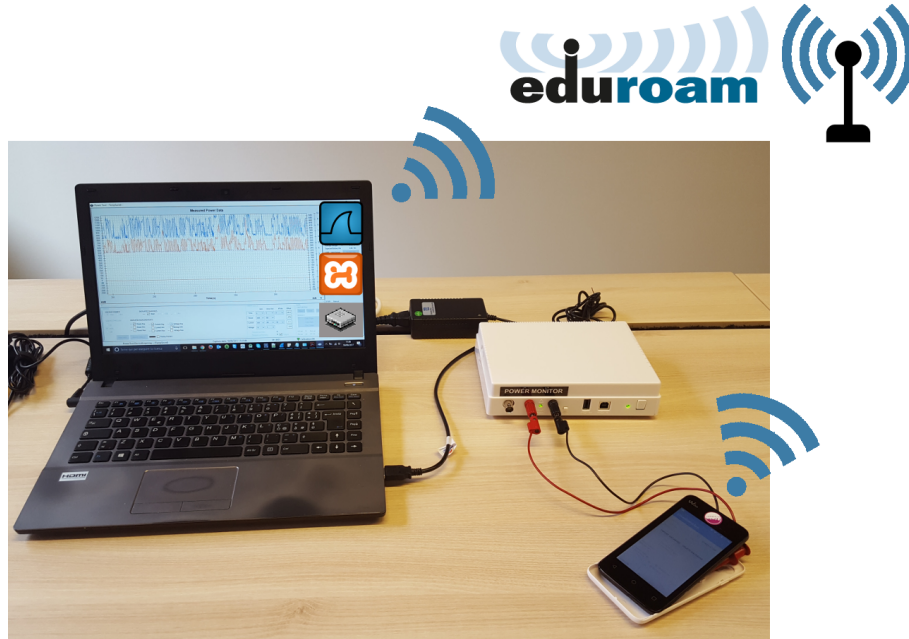


Figure 3.4. Screenshot of power monitor measurements

through the utilization of a power monitor, similarly to previous research [62, 37]. Figure 3.4 shows the setup for the measurements performed with the smartphone under analysis, which is a Wiko Sunny[66]. In the experiments, the smartphone runs Android Marshmallow version 6.0 (API Level 23) and is equipped with a quad-core 1.3 GHz Cortex-A7 processor and has 512 MB of RAM. The smartphone also features 5 MP camera and 8 GB of flash storage. The smartphone provides WiFi and Bluetooth connectivity, supporting 802.11 b/g/n and Bluetooth v 4.0 standards respectively. It is powered by a 1 200 mA, 3.7 V battery. In the experiments, the application performs sensing from GPS, accelerometer and proximity sensors of the smartphone. The smartphone delivers data to a laptop that acts like the cloud collector and also run the software to analyse the power measurements and network traffic (i.e., Wireshark). Since the computational power and the hardware of the laptop offer considerably exceeding performance of the mobile device, the hypothesis is consistent. The laptop is equipped with a dual-core 2.6 GHz Intel i5-4210M and 8 GB of RAM; it has a 256 GB Crucial SSD as storage and an Realtek card for WiFi 802.11 b/g/n connectivity. The laptop runs Windows 10 OS at the time of the tests.

To accomplish the power measurement campaign, I utilize the power monitor hardware by Monsoon [44] while existing works acquired power consumption measures via software by means of system calls [3, 51]. Such a different method was chose because the power monitor directly retrieve measurements while ensuring

a higher level of accuracy. In order to collect data, the power monitor needs to power the smartphone directly, hence in the equivalent circuit it substitutes the internal battery. 5 000 samples per second is the sampling rate to record measures in real-time. Users can export readings in a csv file at the end of the measure campaign exploiting a specific software, which also displays a real-time chart of the measurements. Figure 3.5 shows a sample of energy trace captured by the Monsoon power monitor software. In particular, the trace highlights the different phases of a data collection exploiting delayed-DRM. As is possible to notice in the first part of the energy trace, a very important contribution to battery draining is provided by the device monitor. When active, the model mounted on the Wiko Sunny spend in average 160 mA and pushing on the touchscreen generate a current peak of 260 mA. Another peak appears when the WiFi module starts and the connection is setting up in order to report the collected data. As expected, the transmission phase requires more energy respect to the sensing.

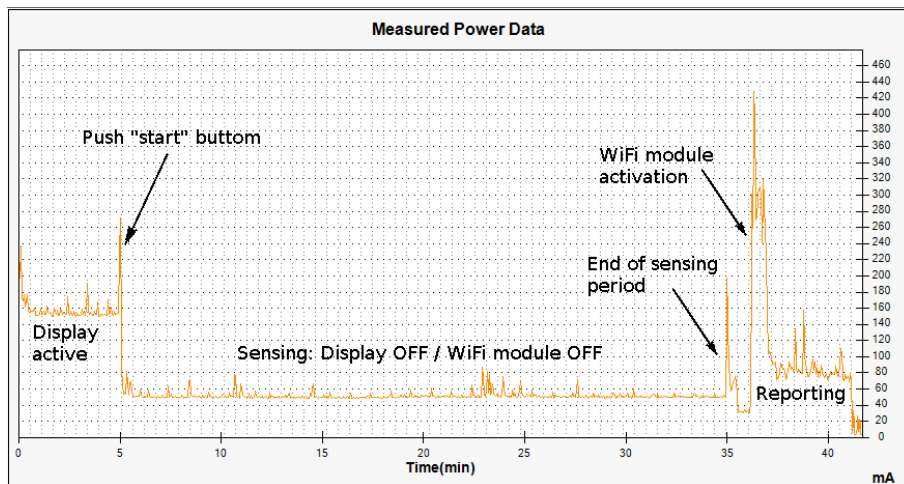


Figure 3.5. Screenshot of power monitor measurements

Wireshark (see Figure 3.6) is an open source network analyser available for all operating systems and employs the GTK+ widget toolkit and pcap library for packet capturing. It is useful to profile network traffic, analyse packets and the structure of different network protocols. Indeed, packet capture can provide information such as transmit time, source and destination, protocol type and header data. Using Wireshark I was able to visualize directly from its panel graphical information about transmitted packets and TCP errors (Figure 3.7). Moreover, it provides export functionalities in order to extract raw data and further elaborate them using third parties applications.



### 3.4 – Measurement Set-Up

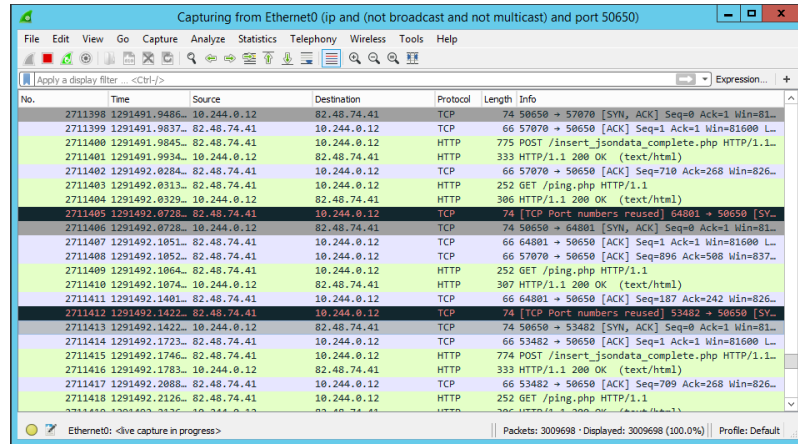
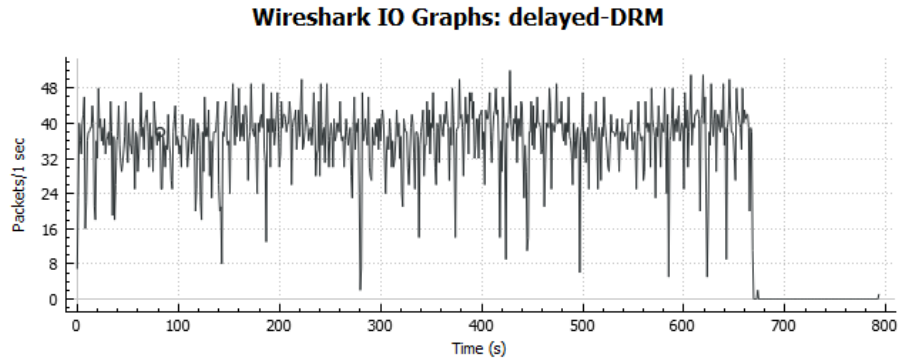
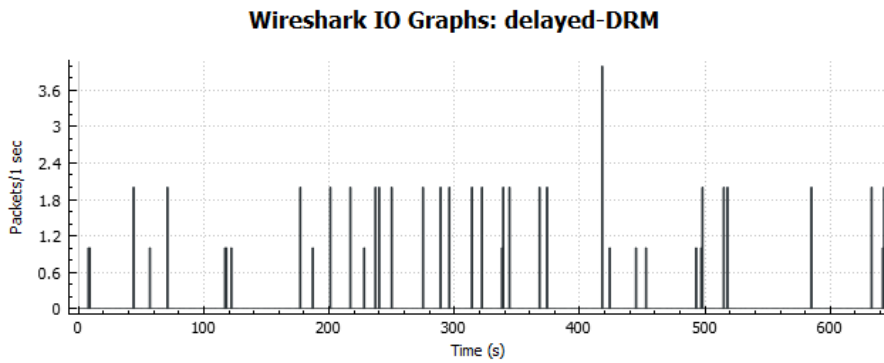


Figure 3.6. Capturing network packets with Wireshark



(a) Packets analysis over time returned by Wireshark



(b) TCP errors analysis returned by Wireshark

Figure 3.7. Graphical representation provided by Wireshark at the end of a test with delayed-DRM

### 3.5 Experimental Results

PARAMETER	VALUE
Sensing time	30 min.
Maximum sample delay	SENSOR_DELAY_NORMAL
Location upload interval	20 sec.
GPS priority	PRIORITY_HIGH_ACCURACY

Table 3.2. Application parameters

This Section presents the results about energy and network measurements obtained from the power monitor and Wireshark. Table 3.2 summarise the values of all the application parameters set to perform tests. Figure 3.8 shows an example of raw energy traces. The comparison highlights the typical differences of each data reporting mechanism. In Figure 3.8(a) is possible to note the two different phases of a delayed-DRM in which the smartphone performs sensing and reporting while in Figure 3.8(b) the continuous-DRM perform both sensing and reporting until the end of the data collection. Moreover, in Figure 3.8(c) and Figure 3.8(d) is possible to note the average current variation while performing continuous-DRM with different values of the threshold  $\delta$ .

In order to obtain a meaningful comparison among the results is necessary to perform measurements considering a common reference point among the data reporting mechanisms. Specifically, different DRM spend different times for reporting, i.e., continuous-DRM keeps active the interface all the time, while with delayed- and probabilistic-DRM, the time in which the interface remains active is supposed to be shorter. For such a reason, the measurement campaign considers for all data reporting mechanisms the same amount of time spent for sensing, i.e., 30 min. Consequently, all the data reporting mechanisms gather the same amount of information that needs to be delivered. However, each data reporting mechanism spends different amounts of time for data reporting. For example, delayed-DRM performs sensing for 30 min and then it takes other 6 min to perform reporting. Figure 3.5 shows an example of measurement profile of the application for delayed-DRM and highlights the various phases of the data reporting mechanism in each point in time. The probabilistic-DRM works with a feedback from the collector  $\delta$  set to 0.5 for comparison with other schemes delayed- and continuous-DRM, but in order to better understand the behavior of such data reporting mechanism additional tests exploit different values of  $\delta$ , namely  $\delta = [0.25, 0.5, 0.75]$ . All the measurements are performed with the EDUROAM WiFi network which allows to test the performance of the data reporting mechanisms in realistic environments while setting-up an ad-hoc network for the tests would

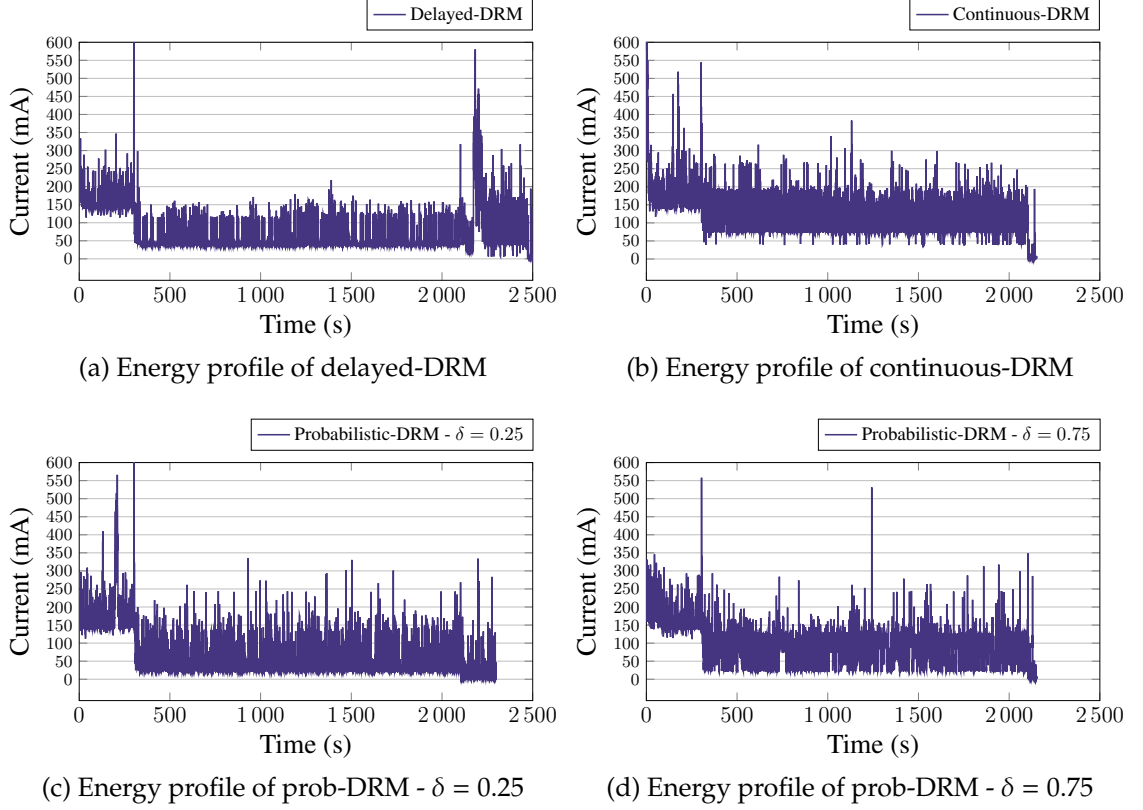


Figure 3.8. Comparison of energy traces of different data reporting mechanisms

have achieved better results but it is not well representative as the environment does not suffer problems such as interference, traffic congestion and in general, the necessity to share available resources.

Figure 3.9 shows the CDF of the battery drain. In particular, Figure 3.9(a) focuses on the various data reporting mechanisms while Figure 3.9(b) focuses on the probabilistic-DRM mechanism and highlights the impact of  $\delta$ , the probability of transmission in each timeslot. Note that each timeslots is worth 40 s and the value of current on the x-axis refer to the peak value measured at a given time. With continuous-DRM, the value of current drained by the battery is low than 75 mA for a significant fraction of the time. Compared to delayed-DRM, continuous-DRM exhibits on average with higher instantaneous peak values. During 50% of the reporting time, while continuous-DRM exhibits peak values above 150 mA, delayed-DRM achieves values above 40 mA. Such behaviour is expected from the theoretical results on WiFi energy consumption as one of the substantial components to the total energy budget depends on the traffic load [58]. On the one hand, continuous-DRM maintains the interface active for longer but transmitting

few packets each time. On the other hand, delayed-DRM sends higher bursts of packets during shorter time periods. probabilistic-DRM exhibits an intermediate behaviour than continuous- and delayed-DRM. When analyzing probabilistic-DRM in Figure 3.9(b), the higher the value of  $\delta$ , the higher the probability to transmit. This translates in a different distribution of the instantaneous peak values of current: with  $\delta = 0.75$ , the peak values are below 75 mA for 75% of the reporting time, with  $\delta = 0.5$  this figure becomes 62.5% of the reporting time and drops to 48% for  $\delta = 0.25$ . Figure 3.10 shows the CDF of the packet rate

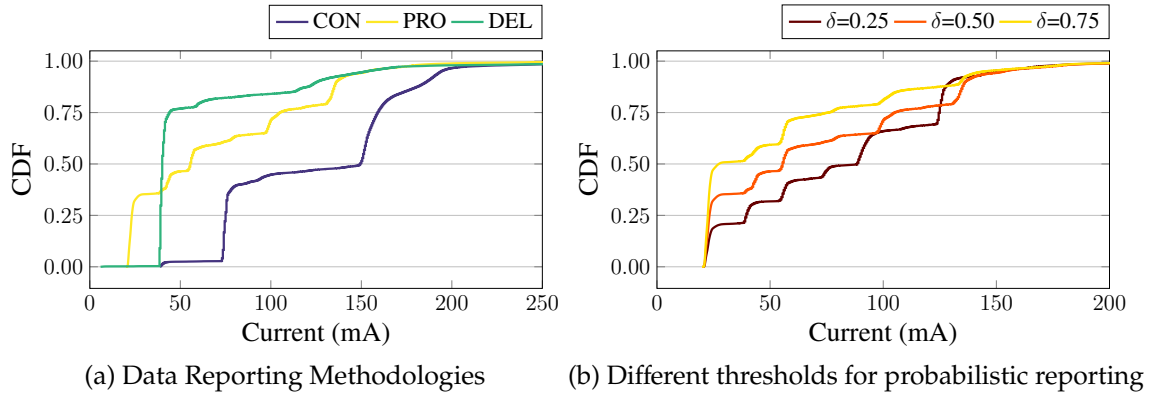


Figure 3.9. CDF of energy spent for different reporting approaches

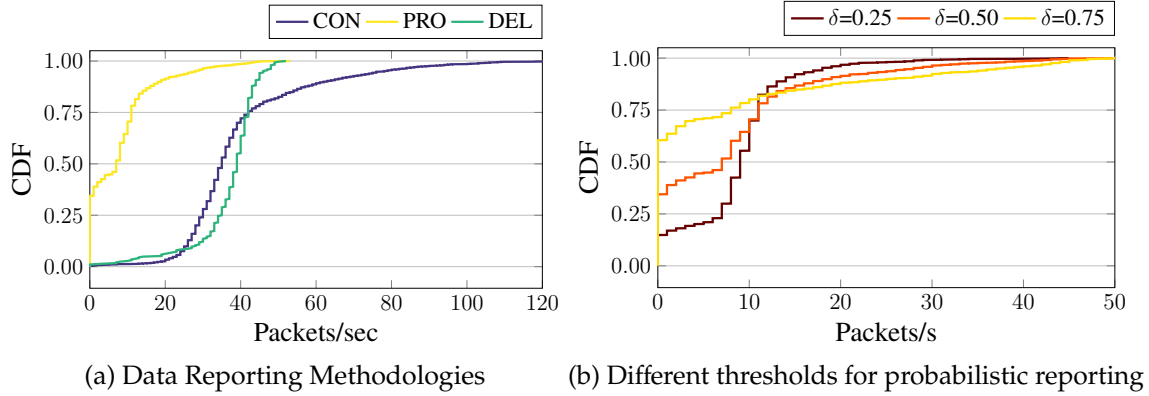
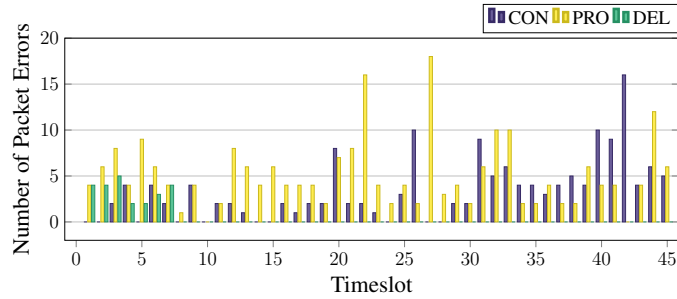


Figure 3.10. Distribution of transmission rates

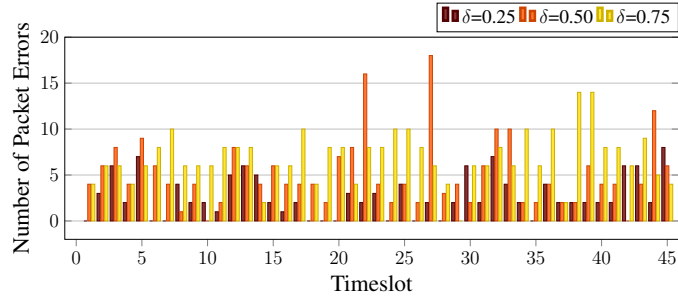
transmission obtained with Wireshark. Figure 3.10(a) compares the data reporting mechanisms that exhibit substantially different distributions of packet transmission rates. Indeed, for 75% of the reporting time, probabilistic-DRM achieves rates as high as 10 packets/s. Compared to probabilistic-DRM, both continuous- and

delayed-DRM achieve rates as high as 40 packets/s for 75% of the reporting time. Interestingly, unlike probabilistic- and delayed-DRM that converge to a certain maximum rate, continuous-DRM has higher variability and packet rates can be as high as 120 packets/s. The reason is due to the technical implementation. While delayed-DRM is basically transmitting a unique (and bigger) file, continuous-DRM transmits frequently smaller chunks. Hence, the collector and the mobile phone have to synchronize much more often to ensure reliable data delivery. Similarly to the result obtained for the energy (see Figure 3.9(b)), Figure 3.10(b) shows that also the transmission rate varies with the increase of  $\delta$ . Interestingly, the variation is evident only for low rates. For example, rates up to 5 packets/s occurs for 20%, 40% and 70% of the reporting time for  $\delta = 0.25$ ,  $\delta = 0.5$  and  $\delta = 0.75$  respectively.

Figure 3.11 shows the distribution of packet errors along time. In particular,



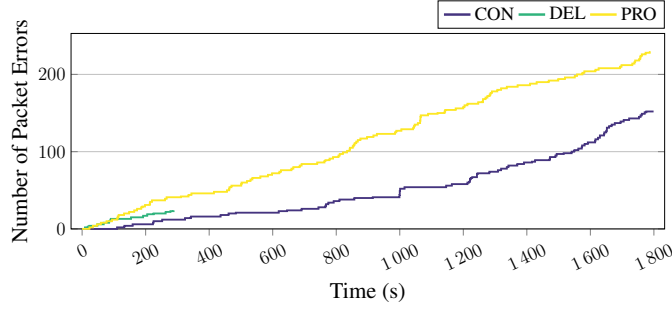
(a) Data Reporting Methodologies



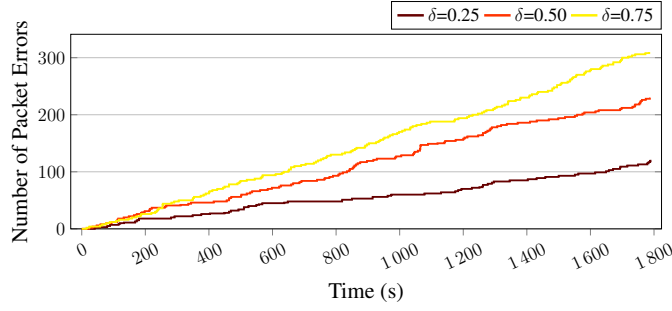
(b) Different thresholds for probabilistic reporting

Figure 3.11. Distribution of packet errors

in order to compare the number of errors occurs during each transmission slot of the probabilistic-DRM, also continuous- and delayed-DRM time periods are subdivided in time slots of equal length (i.e., 40 s) and the errors occurred in each seconds are grouped together. Figure 3.11(a) compares the data reporting mechanisms. As expected, the distribution of packet errors for delayed-DRM is concentrated as the data deliver is shorter. Continuous- and probabilistic-DRM exhibit high variability in packet errors that is due to the realistic environment as



(a) Data Reporting Methodologies



(b) Different thresholds for probabilistic reporting

Figure 3.12. Errors increasing respect time.

that the EDUROAM WiFi network is not under control. However, it should be noted that probabilistic-DRM experiences a comparatively higher number of losses. Indeed, the 802.11 protocol is well known to be inefficient as its scheduling strategy is to allocate single resources to single nodes [63]. Hence, it favors continuous rather than probabilistic data reporting mechanism types of data transmissions. Figure 3.12 confirms last considerations showing the rising of the number of packet errors over time, in Figure 3.12(a) is clear as already mentioned that probabilistic-DRM suffers an higher occurrence of transmission errors respect to continuous-DRM. Moreover, varying the value of  $\delta$ , in Figure 3.12(b) is possible to see the increasing/decreasing of the total amount of errors occurred in probabilistic-DRM. In particular, only with  $\delta = 0.25$  probabilistic- and continuous-DRM exhibit a similar behaviour.

## Chapter 4

# Measurements-Aware Simulations: Results and Analysis of Data Collection Frameworks

After discussing how to develop a mobile crowdsensing application and set-up a measurement environment in order to profile the energy consumption of a smart device while performing data collections based on different data reporting mechanisms (DRM), this chapter presents the analysis of the three data collection frameworks (DCFs) considered. As MCS systems require large users participation to be effective, performing experiments exploiting real resources is often not feasible. For this reason simulation represent a valid alternative. The first Section introduce the methodology employed to realise a large-scale analysis based on simulations in citywide scenario. Hence, Section 4.2 shows the simulation results and compare the effectiveness of the DCFs.

### 4.1 Large-scale Analysis: the Methodology

This section starts with the explanation of how CrowdSenSim simulator works, such a tool was used to obtain large-scale energy traces and information about amount of data collected for each DCF considered moving the users in three cities which exhibits different characteristics such as their size and urban morphology. Specifically, Subsection 4.1.2 shows how to use the simulator in order to obtain the necessary information to perform the data collection frameworks evaluation. Next, Subsection 4.1.3 explains the setting of the simulator.

### 4.1.1 CrowdSenSim

CrowdSenSim is a custom simulator specifically design at the University of Luxembourg to assess the performance of crowdsensing activities in large urban areas [15]. CrowdSenSim supports pedestrian mobility in citywide scenarios where all the individual walking paths are known before simulation runtime to ensure scalability of the platform. The simulator is built with independent modules can inter-operate one with each other and allows to perform different crowdsensing-related analysis such as user rewarding, task allocation or fairness between participants. In CrowdSenSim, the layout of the city is defined in terms of a set of coordinates  $C$  containing information on  $\langle \text{latitude}, \text{longitude}, \text{altitude} \rangle$ . The set of coordinates compose the street network of the urban area where the participants will move during runtime. They are obtained through the combination between an algorithm specifically developed running in the background of CrowdSenSim and OSMnx [5]. OSMnx is a Python package that allows the researchers to easily download and analyse street networks of any location in the world from OSM, which is a map created by people under an open license. Unfortunately, OSM street nodes are inconsistent for direct use in CrowdSenSim because they include dead-ends, intersections and all the points in a segment when the streets curve. OSMnx automatically simplifies and corrects the street topology through an algorithm by removing those points and unifying each resulting set of sub-edges into single edges. However, the resulting topology still lacks of a sufficiently fine-grained level of detail. Hence, CrowdSenSim runs in the background an algorithm that augments the precision of the OSMnx topology by adding nodes on the streets with user-defined level of detail (e.g., 1 meter).

### 4.1.2 CrowdSenSim for Data Collection Frameworks evaluation

To assess the energy consumption for a sensing campaign in real urban environments, we include in CrowdSenSim the real energy-traces for sensing and reporting and the amount of data collected by the specific smartphone used for the test. Such traces are obtained through the measurements on the Android application, the smartphone (i.e., Wiko Sunny) details are described in Section 3.4. Specifically, the cost that each device experiences is computed proportionally to the time of contribution. The reference power consumption profiles and amount of data were obtained from 30 min long sensing traces (see Figure 3.9(a)).

For the evaluation, the ParticipAct dataset was exploited to establish the user arrival pattern in the simulator. In particular, the simulator uses the profile of the average number of contacts during 7 days. The ParticipAct dataset was generated by a real MCS campaign of approximately 170 students in the Emilia Romagna region (Italy) [9]. Specifically, the total simulation period is divided into hours, estimating the minimum number of individuals to be allocated to have an average



user contact following the ParticipAct profile. By definition, a unique user contact is the overlap within a timeslot of two user walking paths so that their distance is below a radius  $R$ . For an implementation choice, multiple overlaps in different timeslots count as distinguished contacts, while multiple overlaps in the same timeslot are considered unique contacts.

The implementation of the DCFs is based on the energy consumption model previously described in Section 2.2 and briefly explained below. Users contribute data according to the exploited DCF during runtime and the simulator computes the amount of gathered data for each user and the associated battery drain for sensing and reporting [6]. Moreover, the evaluation needs to consider another important aspect necessary to stimulate participation. The fairness among the users.

The next paragraphs explain how the three data collection frameworks and the fairness index were implemented. Hence, The simulation results are presented in Section 4.2 and help to analyze the data reporting mechanism and how much employed crowdsensing techniques are effective.

### Implementation of Data Collection Frameworks

In every data collection frameworks the working time is subdivided in timeslots. The simulator implements a different decision strategy for each timeslot in order to allow or not data generation and reporting. The computation of the energy consumption is performed deriving from the energy-traces a value in mA per minute both for the sensing and reporting phases of each DRM. Moreover, the total amount of data collected was estimated extracting a value in KB per minute based on the real total amount of data collected during 30 minutes.

Deterministic Distributed Framework (DDF) allows both data generation and reporting in each timeslot until the battery percentage of the device decrease of 1%. When the energy consumption reach this threshold the device stop to perform data collection.

The Probabilistic Distributed Algorithm (PDA) continuously sensing while reporting is driven by a probability randomly generated in the range [0-1] in each timeslot. In order to allow or not data reporting, the probability is checked against a threshold which indicate the feedback provided by the collector. The threshold varies with the quantity of data already send following the equation:

$$\delta = 1 - (SI)^{0.5}, \quad (4.1)$$

where:

$$SI = \frac{\text{Total delivered data}}{\text{Total expected data}}. \quad (4.2)$$

The total expected data depends on the generated data in DDF case and vary with the increasing of users. If the probability of sending data is higher than the

threshold the reporting is enable. On the contrary, if the probability is smaller the sensed data are stored until the first reporting slot.

Piggybacking Crowdsensing (PCS) is similar to the PDA except for the definition of the threshold and the amount of the consecutive timeslots in which is able to reporting. Indeed, in this case the threshold varies during the day following a calls arrival distribution model. Furthermore, when a slot start to send, the reporting is allow for a number of consecutive slot defined by the distribution of average calls duration model. Both the model are presented in [67].

## Fairness

Ideally, in MCS systems the collector should guarantee fair treatment to each of the participants, i.e., it should not take advantage from a small set of users. Intuitively, users that contribute higher amounts of data, sustain a higher energy cost to produce such data and therefore need to be rewarded adequately.

To evaluate fairness between users in a DCF, we exploit the Jain Fairness Index [29]. The index measures the equality in allocating a set of finite resources to users according to certain criteria. For instance, if a user is willing to pay a sum that is twice more high than other users for a shared resource, the scheduling algorithm should allocate to him twice the amount of resources than the other users. In MCS context, we interpret this index as a way to measure the fairness in which a set of users contribute to create a finite set of resources, i.e., the total amount of data delivered to the collector. Specifically, the index measures the degree of equality in which each DCFs make users to produce their individual amount of data and the associated energy cost. As a first step, separate indexes are defined for each of the two components and then is introduced the definition of a global fairness index to incorporate both components.

Intuitively, users that walk for longer time periods are expected to contribute more data than others. We define the data contribution fairness index ( $F_D$ ) as follows:

$$F_D = \frac{\left( \sum_{i=1}^N d_i \right)^2}{N \cdot \sum_{i=1}^N d_i^2}, \quad (4.3)$$

where:

$$d_i = \frac{D_i}{D_i^M}. \quad (4.4)$$

$D_i$  is the amount of contributed data from user  $i$  and  $D_i^M$  is the maximum amount of information an individual could contribute in the corresponding time.  $F_D$  assumes values equal to 1 when users contribute data proportionally to the time spent walking. However, this index presents a significant drawback if considered alone.

Indeed,  $F_D$  does not distinguish between two users that walk for the same time period, but have different initial levels of battery. For this reason, we introduce an additional index that takes into account the battery level of the devices. Specifically, a device with a higher battery level prior to the start of the sensing process is expected to contribute higher amounts of data than devices with a lower battery level. The index of battery fairness index ( $F_B$ ) is defined as:

$$F_B = \frac{\left(\sum_{i=1}^N b_i\right)^2}{N \cdot \sum_{i=1}^N b_i^2}, \quad (4.5)$$

where:

$$b_i = \frac{B_i}{B_i^T}. \quad (4.6)$$

$B_i$  and  $B_i^T$  are measurements of battery level of the mobile device  $i$  (in mAh). The former is the amount of *battery drain* experienced during the contribution process and the latter is the total battery when the mobile device starts to contribute data.

We introduce the crowdsensing fairness index ( $F_{CS}$ ) to simultaneously take into account both the battery drain and the amount of contributed data. Specifically:

$$F_{CS} = \sigma \cdot F_D + (1 - \sigma) \cdot F_B, \quad (4.7)$$

where  $\sigma$  is a balancing coefficient that assumes real values in  $[0,1]$  and weights the relative importance between the two indexes  $F_D$  and  $F_B$ .

### 4.1.3 Simulation Setting

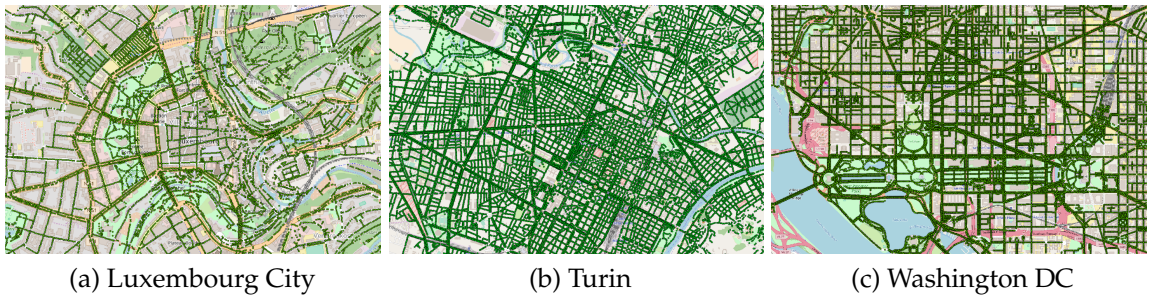


Figure 4.1. Cities taken into account for the evaluation. The green dots represent the pedestrian lanes where users walk obtained with an algorithm running in the background of CrowdSenSim

The evaluation is performed considering three different cities. Figure 4.1 shows the selected cities where users can move during simulation runtime: Luxembourg City (see Figure 4.1(a)), Turin (Italy) (see Figure 4.1(b)) and Washington DC (USA) (see Figure 4.1(c)). They have been chosen on purpose for two reasons. The first is that the selected cities can be ordered by growing size. The center of Luxembourg City covers an area of 51.47 km<sup>2</sup> with a population of 114 090 inhabitants as of the end of 2016 and is a fast growing city with headquarters of many international institutions (see Figure 4.1(a)). The city center of Turin occupies an area of 130.17 km<sup>2</sup> and has a population of 883 601 inhabitants as of the beginning of 2016 (see Figure 4.1(b)). The city center of Washington DC covers approximately an area of 158.1 km<sup>2</sup> with a resident population of 672 228 inhabitants as of the end of 2015 (see Figure 4.1(c)). The second motivation behind the choice is about the urban morphology, which determines the street network topology. Luxembourg City shows the common north european urban morphology with many short streets with small lanes, a high density of crossroads in the center and few parallel large streets in the periphery. Washington DC is totally different from Luxembourg City and its topology of street network presents large lanes with a high number of parallel long streets. In addition, the differences between the urban morphology in the city center and the periphery are minimal. Turin falls in between the two former categories because of typical roman grid street organization. As explained in Section 4.1.1, the user arrival pattern exploited in CrowdSenSim is based on realistic mobility traces and chosen simulation period is 12 consecutive hours in one day. The PartecipAct dataset supplies information on the user contact per-hour. Following this idea, the simulator assigns a certain number of participants to arrive at the desired sum of contacts for each hour. In detail, we consider a user contact when two walking paths are within a certain timeslot, so that two individuals are in a radius of 50 m. Only one mobile device for each user generates data. The users walk with an average speed uniformly distributed between [1,1.5] m/s in a period of time that is uniformly distributed between [1,40] minutes in all the simulations based on theoretical data obtained by datasheets and [20,40] minutes for the simulations exploiting real data obtained by energy measurement. Participants move over a street network in a random walk fashion between a random generated starting and arrival point respecting the walking time constraint. The number of users is also set differently like the time of walking and is equal to 20 000 and 10 000 respectively, unless otherwise stated. The current battery charge is generated considering a value of full capacity and an initial level. The full battery capacity is variable following most popular models of smartphones available on the market and is randomly picked from a list including 2200 mAh (Huawei P8 Lite), 2550 mAh (Samsung Galaxy S6), 2800 mAh (LG G5) and 3300 mAh (Samsung Galaxy J7). Also the initial battery level is different in the two simulation case and it is uniformly distributed in the range [80-90]% with

theoretical data and [10-90]% with real data.

SENSOR	PARAMETER	VALUE	UNIT
Accelerometer (MPU-6500)	Sample rate	4	kHz
	Sample size	6	Bytes
	Current	450	$\mu$ A
Proximity (TMD4903)	Sample rate	8.1	MHz
	Sample size	2	Bytes
	Current	150	$\mu$ A
GPS (SKG13BL)	Update period	10	s
	Sample size	24	Bytes
	Current	23	mA

Table 4.1. Sensors parameters

In theoretical simulations, data generation takes place exploiting heterogeneous sensing equipment commonly available in today mobile devices, including the MPU-6500 3axis linear accelerometer from InvenSense, the TMD4903 proximity sensor from AMS and the SKG13BL GPS module from SKYLAB. Communications occur over the WiFi link, having obtained the precise location of WiFi hotspots in form of  $\langle \text{latitude}, \text{longitude} \rangle$ . In the simulator, the time is slotted. The maximum quantity of data that can be sent in a time slot is defined by the transmission speed, fixed to 1 Mbps. Table 4.1 presents the detailed information on the sensor equipment.

DRM	WORKING PHASE	AVERAGE CURRENT ( $mA$ )
Delayed	sensing	52.29
	reporting	112.45
Continuous	sensing & reporting	125.32
Probabilistic	$\delta = 0$	33.67
	$\delta = 0.25$	56.00
	$\delta = 0.50$	70.61
	$\delta = 0.75$	81.56
	$\delta = 1$	132.72

Table 4.2. Average current spent in different data reporting mechanisms

For simulations with real energy measurements instead, the energy consumption of different data reporting mechanism is shown in Table 4.2. As is possible to

see, the sensing phase of delayed-DRM and probabilistic-DRM with  $\delta = 0$  differ of approximately 20 mA. This value is probably due to the different implementation on the Android application. Note that delayed-DRM exploit a local database while probabilistic-DRM a simple buffer to store the gathered data.

## 4.2 Simulation Results

This section presents the results obtained exploiting CrowdSenSim. In each Subsection, we first evaluate the distribution of the energy consumption for the three Data Collection Frameworks (DCF) in Luxembourg City. Then, we investigate the performance of the DCFs for various cities and show for a limited number of users the active contribution periods to highlight the differences between the DCFs. Finally, we assess the amount of collected data.

### 4.2.1 Simulations Based on Datasheets Informations

The first Subsection is focused on the simulations where energy and data parameters was extracted from the datasheets of hardware commonly used in smart devices (see Table 4.1).

#### Energy Consumption

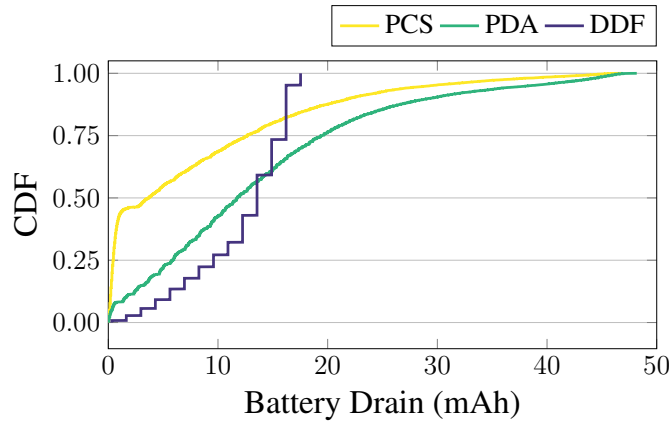


Figure 4.2. CDF of battery drain per user for different DCFs in Luxembourg City. Simulations based on commonly used hardware datasheets

The first comparison is related to the energy spent during data collection. Figure 4.2 presents the CDF of the per-user battery drain for the considered DCFs in Luxembourg City. By design, DDF includes a stopping mechanism to prevent

users contribute additional data upon meeting given criteria, such as if the battery drain attributed to previous sensing and reporting operations has exceeded a given threshold or if the amount of previous contributed data has reached a certain value. Hence, it limits the maximum energy consumption the users spend, i.e., in this experiment all the users spend at maximum 17 mAh. Comparatively, the percentage of users that spend more than 17 mAh is 20% and 30% for with PDA and PCS respectively. Interestingly, DDF lowers the number of users with low energy consumption. This means that the organizer of the sensing campaign effectively exploits the users that agreed to participate and contribute data and that are compensated for such contribution. On the contrary, a significant fraction of users consume a little amount of energy with PCS and PDA. The reason is the probabilistic data delivery mechanism that if applied for periods of time in the order of hours prevents some of the users to transmit significant amounts of data. Note that in the context of crowdsensing, the dominant factor affecting energy consumption is data delivery and not sensing [6]. The data collection

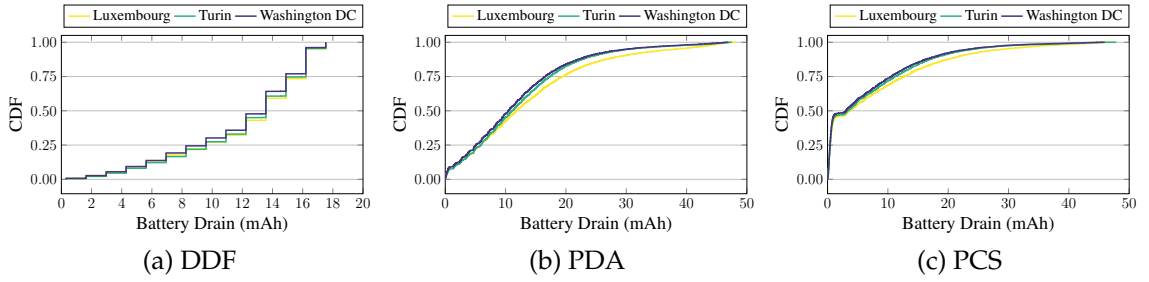


Figure 4.3. CDF of battery drain per user for considered DCFs in different cities. Simulations based on commonly used hardware datasheets

frameworks were tested in three different cities in order to evaluate the impact of different sizes and urban morphology. In Figure 4.3 is shown CDF of battery drain per user with considered DCFs in Luxembourg City, Turin and Washington DC. Interestingly, the various DCFs behave similarly within the same city and the minor variation is attributed to Washington DC. Consequently, the size of the city has a minor impact on the performance of the DCFs. Note that DDF exhibits a CDF that mimics a step function. Each step identifies the group of users that stopped contributing data because of the stopping mechanism and have delivered to the system a similar amount of data. Figure 4.4 shows the amount of collected data and the associated battery drain for all the DCFs. Each mark of the plot represents the energy consumptions that a set of users has spent to produce a given amount of data. First, it should be noted that DDF exhibits a low number of marks. The reason is that the users exhibit a similar behaviour as DDF indirectly controls the level of energy consumption. On the other hand, the other DCFs

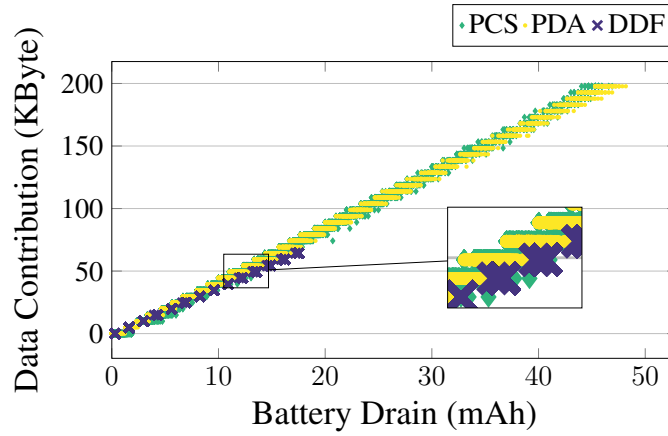


Figure 4.4. Amount of collected data and the associated battery drain. Simulations based on commonly used hardware datasheets

exhibit much higher variability due to the probabilistic reporting: to produce the amount of data, users spend a different amount of energy. This variability becomes higher as the total amount of data increases. Practically, the result shows that providing user rewards on sole basis of the amount of contributed data fails to properly compensate for users' costs because of the technical implementation of data reporting.

### Amount of Collected Data

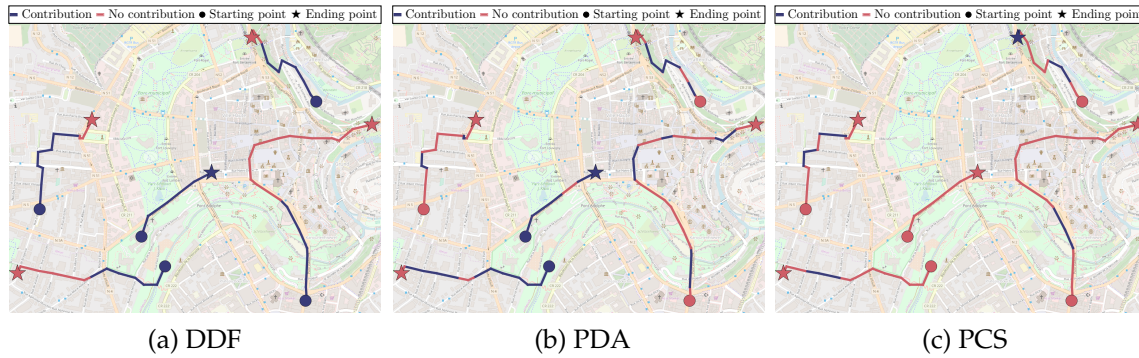


Figure 4.5. User trajectories with the associated data contribution in Luxembourg City

Here is compared the data contribution provide by each DCF. In Figure 4.5 is possible to see the trajectories of five users walking in Luxembourg City that contribute data with the various DCFs. The objective is to highlight the active



periods of contribution to clearly show the differences between the reporting mechanisms. With DDF, data contribution is continuous until users stop sending data because of the sufficient amount of contribution. With PDA, users generate data in an intermittent fashion depending on the probability. PCS shows that a user can also not contributing in case during the walking period no calls or applications are exploited. Figure 4.6 shows in form of heatmap the spatial distribution of the

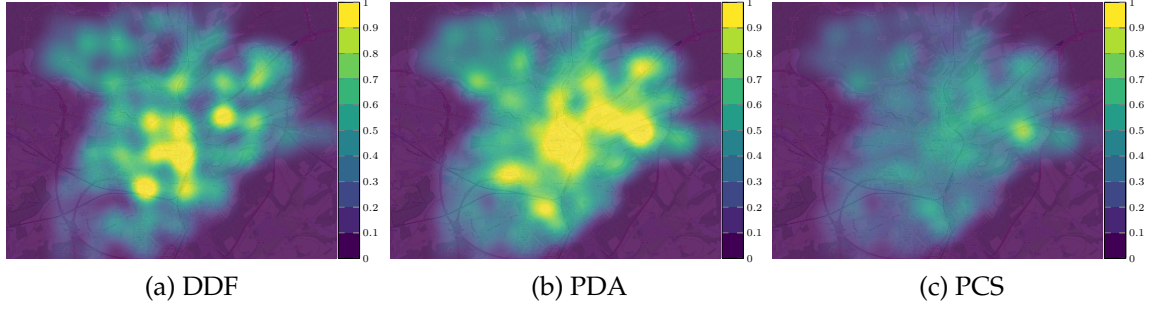


Figure 4.6. Normalized distribution of amount of contributed data in Luxembourg City comparing different DCFs. Simulations based on commonly used hardware datasheets

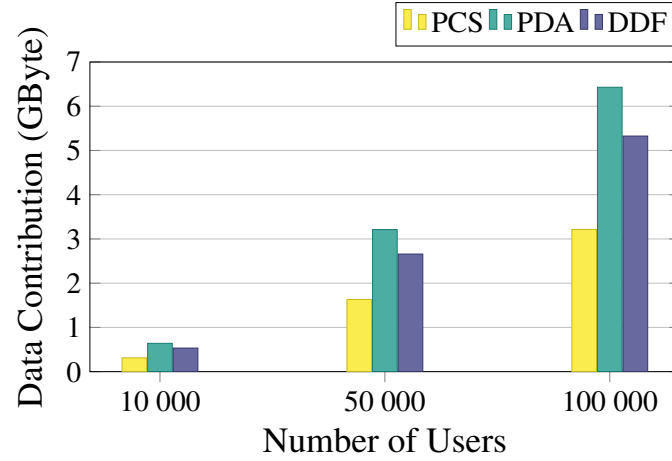


Figure 4.7. Amount of contributed data for considered DCFs in Luxembourg City. Simulations based on commonly used hardware datasheets

total amount of collected data at the end of the simulation period for Luxembourg. The heatmap is normalized between 0 and 1, where 1 indicates a total of 10 MB of data generated during the entire simulation period. PDA achieves the higher spatial distribution of amount of collected data than the others DCFs. This is

because it does not include any mechanism to stop contribution. DDF shows a lower amount of collected data due to the stopping mechanism, permitting energy savings as shown in Figure 4.2. PCS achieves the lowest amount of contributed data. Indeed, although users perform continuous sensing, data reporting fully depends on the probability of performing calls. Figure 4.7 shows the amount of collected data in Luxembourg City comparing the considered DCFs for different number of users. PDA is the DCF that contributes the highest amount of data, as users are not prevented by any stopping mechanisms. DDF presents a big amount of data in the first phases due to continuous reporting, but then users stop to save energy. On the contrary, PCS achieves the lowest amount of data collected and fails to capture area of interests with particular accuracy. Again, the motivation lies in the reporting mechanism implemented.

#### 4.2.2 Simulations Based on Measured Energy Consumption and Data Generated

This Subsection shows the results obtained feeding the simulator with the real energy-traces and amount of data collected during the tests performed exploiting the Android application, the power monitor and Wireshark.

##### Energy Consumption

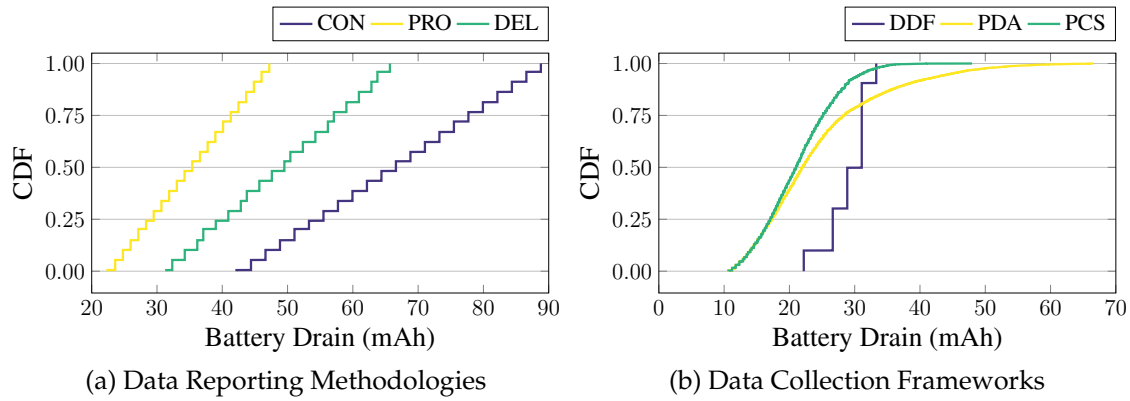


Figure 4.8. CDF average battery drain per user on large scale. Simulations based on real energy-traces and amount of generated data

The real traces allow to evaluate the energy consumption exhibits during data collection. Figure 4.8 presents the CDF of the per-user battery drain for the proposed DRMs and the DCFs under analysis in Luxembourg City. Interestingly,

the difference between DRMs and DCFs is substantial. The reason is that DRMs lack of important components of DCFs such as a feedback from collector on data utility or a criteria to stop contribution. Figure 4.8(a) shows the CDF of battery drain for the DRMs, which differ by range of values and slope. The steps in the profiles represent groups of users that have achieved a certain amount of battery drain. As expected, CON is the most energy consuming DRM and exhibits the highest range variability while PRO and not DEL is the less consuming DRM. However, implementing expensive DRM like CON in a DCF that can be tuned to limit user contribution like DDF is beneficial (see Figure 4.8(b)). With DDF, all the mobile devices spend 33 mAh at maximum. In comparison, the percentage of users that spend more than 33 mAh for with PCS and PDA is respectively 3% and 16%. DDF reduces the percentage of mobile devices with low battery drain. This fact highlights that the organizer of a campaign employs effectively participants that accepted to join the campaign for gathering information and are rewarded on this basis. On the other hand, a substantial number of devices consume a little amount of battery with PCS. With PDA, most users consume a bigger amount of energy compared to PCS, but it is due to an increase of the amount of contributed data, as will be later highlighted in Figure 4.11 and in Figure 4.12.

Another interesting aspect to assess is the impact of performing a data collection framework in different cities. Figure 4.9 illustrates the similarity of battery drain across different urban environments, showing that the urban morphology of street networks and the size of the city has a minor impact on the performance of the DCFs. DDF exhibits a CDF that mimics a step function, where each step indicates a group of participants that has contributed to the collector a similar amount of data. Hence, they have stopped to acquire data for one of the possible motivation: (i) the battery drain due to sensing and reporting operations has exceeded a given value, (ii) users already reported a fixed amount of data. Figure 4.10 presents the

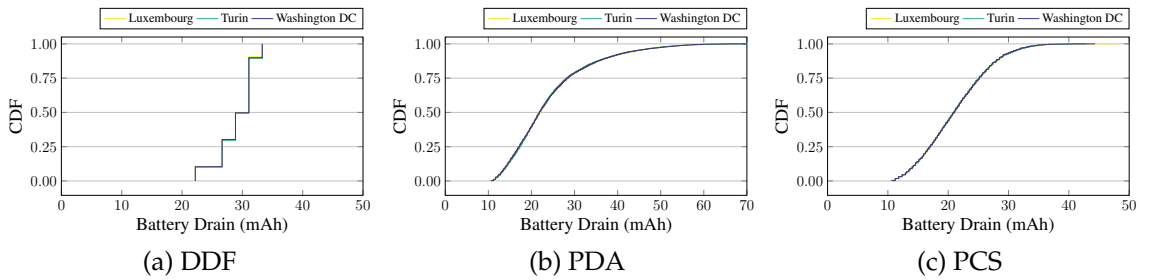


Figure 4.9. CDF of battery drain per user for considered DCFs in different cities. Simulations based on real energy-traces and amount of generated data

amount of reported information and the associated energy consumption for each DCF. Marks in the graphic represent the battery drain that a group of participants

has consumed to contribute a certain amount of data. Interestingly, DDF presents a low number of marks. The motivation is that mobile devices show a similar behavior because of the stopping mechanism which indirectly controls the energy consumption. On the other side, PDA and PCS show much higher variability due to the different reporting mechanisms: to contribute a certain amount of data, the participants spend different amounts of energy. This variability increases when the total contribution becomes higher. From a practical point of view, simulations exhibit that applying an incentive mechanism based only on the amount of acquired information misses to reward fairly individuals because of the technical implementation of reporting policies.

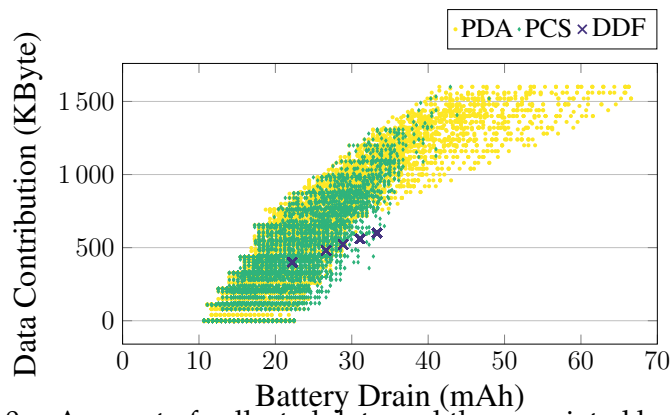


Figure 4.10. Amount of collected data and the associated battery drain in Luxembourg City. Simulations based on real energy-traces and amount of generated data

### Amount of Collected Data

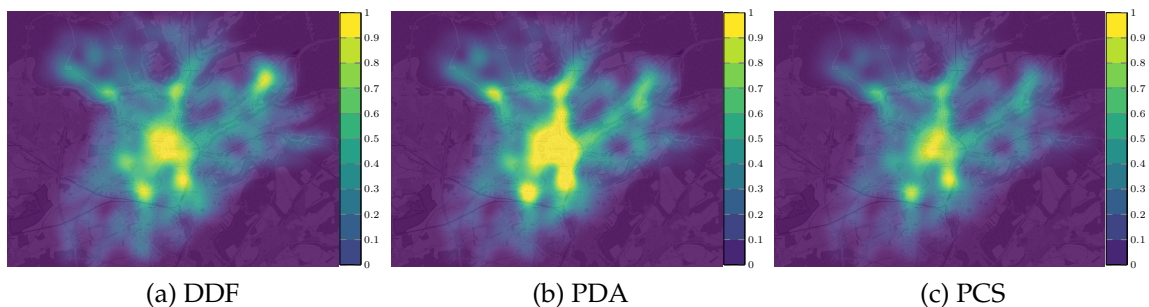


Figure 4.11. Heatmaps of Luxembourg city with different DCFs. Simulations based on real energy-traces and amount of generated data

The measurements provide also a reference value about the quantity of data collected from DRMs. Also such information is used for the evaluation, Figure 4.11 shows the spatial distribution of the total amount of collected data at the end of the simulation period for Luxembourg. The heatmap is normalized between 0 and 1 and 1 indicates a total of 100 MB of data generated during the entire simulation period. PDA achieves a high spatial distribution of amount of collected data and this is because it collects data until the collector has gathered a sufficient amount of data and lowers the transmission probability of the users. DDF shows a lower amount of collected data in the center due to the stopping mechanism as shown in Figure 4.8b. PCS achieves the lowest amount of contributed data. Indeed, although users perform continuous sensing, data reporting fully depends on the probability of performing phone calls. Figure 4.12 shows the amount of

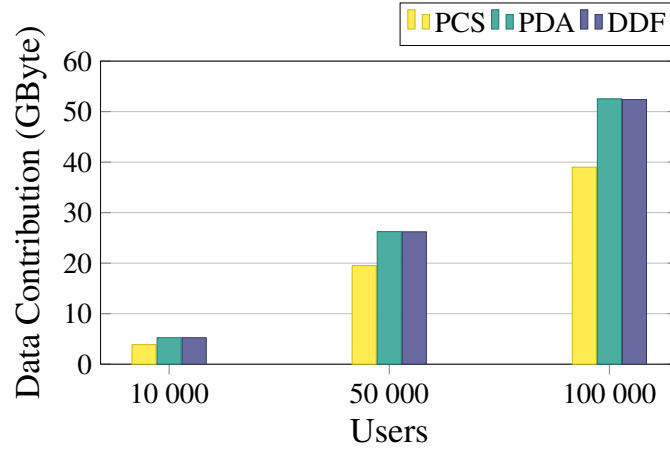


Figure 4.12. Amount of contributed data for considered DCFs in Luxembourg City. Simulations based on real energy-traces and amount of generated data

contribution in Luxembourg City comparing the considered DCFs for different number of users. DDF is able to gather a considerable amount of data due to continuous reporting even if users stop to contribute to save energy. PDA achieves a similar amount of contributed data of DDF. On the contrary, PCS achieves the lowest amount of acquired information and would most likely fail to accurately monitor area of interests. Again, the motivation lies in the reporting mechanism implemented.

### Fairness

The data collection frameworks should always exhibit a fair behaviour among the users. Figure 4.13 compares DCFs according to the different fairness indexes presented in Subection 4.1.2. The results are obtained after 100 rounds of simulations. Figure 4.13(a) shows the data contribution fairness index ( $F_D$ ) in form of boxplots.

DDF achieves the highest values of fairness in data contribution and the reason is twofold. First, feedback from collector regulates the amount of data generated. Second, the contribution for each participant is fair in proportion to the amount of time they walk, i.e., users that walk for longer time periods contribute more data. PCS, that depends on the probability of users phone calls is less fair than DDF. In other words, users do not contribute the same amount of data when walk for the same amount of time. PDA presents the lowest value of data contribution fairness because some of the users contribute significant amounts of data when the collector misses data, but if the urgency for additional data is low the users do not contribute at all. Note that for design choice, PDA implements the AO-S algorithm, which is a basic setting and is not as fair as the AO-F variant [45]. Figure 4.13(b) shows the results on the battery fairness index ( $F_B$ ). If compared to the data fairness index, DDF presents much lower values because of the mechanism that stops users if they have spent a certain amount of energy for data contribution. However, this does not differentiate between users that have high or low initial battery level. For example, upon setting the stopping threshold to 1% of the battery, the impact is different if the initial level is 10% than if it was 90%. PCS and PDA are very similar and lower than DDF. Although none implements a stopping criteria, the implicit stopping method is given by the amount of time the users walk. Finally, Figure 4.13(c) shows the Crowdsensing Index ( $F_{CS}$ ) which combines the previous indexes together. PCS is the most uniform is due to the lack of feedbacks from the collector and sporadic contribution of each mobile device during phone calls. PDA shows a linear decrease between data and battery fairness while DDF is the opposite and exhibits a linear increase. Such behavior is due to the underlying properties of the DCFs.

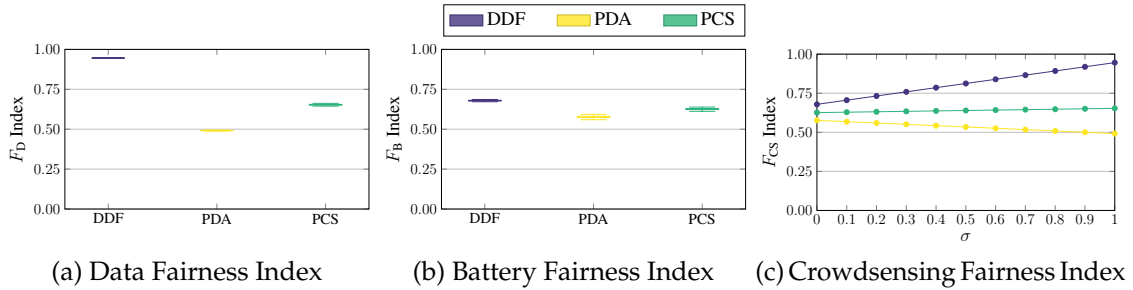


Figure 4.13. Fairness Indexes

## Chapter 5

### Conclusion and Future Works

Since the Data Collection Framework (DCF) define the efficiency of MCS systems this work aims to define a methodology to measure the energy consumption and amount of collected data performed by a DCF and evaluate it in a specific city. Profiling the energy requirement of a data collection framework is crucial to assess the costs of a sensing campaign and to properly plan user incentives plans like monetary rewarding. This work presents the first experimental assessment and comparison of energy consumption of DCFs. An Android application was developed for the purpose and a power monitor and Wireshark was used to obtain energy- and network-traces. The acquired energy measurements have been utilized for large-scale analysis with CrowdSenSim simulator to test effectiveness of the frameworks in Luxembourg, Turin and Washington DC. The results show that DCFs energy consumption differ because of the data reporting mechanism implemented: DCFs with probabilistic reporting comparatively achieve higher energy consumption. Furthermore, such DCFs present high variability, i.e., to produce the same amount of data, the associated energy cost of different users can be significantly different. Consequently, DCFs with continuous reporting that implement mechanisms to block sensing and data delivery after a certain amount of contribution are more effective in harvesting data from the crowd.

The network infrastructure exploited to allow the crowd to send data to the collector will continuously enhanced in order to achieve higher performance. In particular, The next generation of cellular network provide innovative and energy efficient technique such as Device-to-Device (D2D) communications which is very useful to mobile crowdsensing. As an example, considering a certain area of interest, one or more devices may work in a poor signal quality zone and as a consequence reaching the collector may be difficult and expensive in terms of energy spent. However, exploiting D2D communication a nearby device with a better link can act as relay and ensure reliable connectivity to the collector. Moreover, it is possible to deliver messages necessary to manage MCS campaign to the unreachable users. As a consequence they continuing to gather useful

information with the possibility to delivery it at a later time.

D2D communication also allows to design new DCFs where the users group organize themselves in order to optimize the gathering process, avoid data redundancy and consequently the transmission of data already available to the collector. Moreover, D2D techniques provide an important contribution on the system energy efficiency since direct transmission between nearby devices usually occur at a much lower transmit power respect to a wide area communication through mobile base station. Such kind of implementation can considered also the possibility to elect a "central" device in a considered group and use it as a gateway in order to establish only one long distance connection and exploit packet aggregation properties. In such an environment, the implementation of new kind of DCF with local communication may improve the overall system performance so the research needs to deeper investigate their energy efficiency and effectiveness.



# Bibliography

- [3] J. C. Valdivia Bedregal, E. G. C. Gutierrez, and R. E. Arisaca M. "Optimizing energy consumption per application in mobile devices". In: *International Conference on Information Society*. June 2013, pp. 106–110.
- [4] P. Bellavista et al. "LTE proximity discovery for supporting participatory mobile health communities". In: *2017 IEEE International Conference on Communications (ICC)*. May 2017, pp. 1–6. doi: 10.1109/ICC.2017.7996486.
- [5] Geoff Boeing. "OSMnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks". In: *Computers, Environment and Urban Systems* 65 (2017), pp. 126–139. issn: 0198-9715. doi: <https://doi.org/10.1016/j.compenvurbsys.2017.05.004>.
- [6] A. Capponi et al. "A Cost-Effective Distributed Framework for Data Collection in Cloud-Based Mobile Crowd Sensing Architectures". In: *IEEE Transactions on Sustainable Computing* 2.1 (Jan. 2017), pp. 3–16. issn: 2377-3782. doi: 10.1109/TSUSC.2017.2666043.
- [7] A. Capponi et al. "Assessing Performance of Internet of Things-Based Mobile Crowdsensing Systems for Sensing as a Service Applications in Smart Cities". In: *IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*. Dec. 2016, pp. 456–459. doi: 10.1109/CloudCom.2016.0077.
- [8] Daniel Castro et al. "Predicting Daily Activities from Egocentric Images Using Deep Learning". In: *Proceedings of the 2015 ACM International Symposium on Wearable Computers*. ISWC '15. Osaka, Japan: ACM, 2015, pp. 75–82. isbn: 978-1-4503-3578-2. doi: 10.1145/2802083.2808398. url: <http://doi.acm.org/10.1145/2802083.2808398>.
- [9] Stefano Chessa et al. "Mobile crowd sensing management with the ParticipAct living lab". In: *Pervasive and Mobile Computing* 38 (2017), pp. 200–214. issn: 1574-1192. doi: <https://doi.org/10.1016/j.pmcj.2016.09.005>.
- [10] Jiangpeng Dai et al. "PerFallD: A pervasive fall detection system using mobile phones". In: *2010 8th IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*. Mar. 2010, pp. 292–297. doi: 10.1109/PERCOMW.2010.5470652.

- [11] Jakob Eriksson et al. "The Pothole Patrol: Using a Mobile Sensor Network for Road Surface Monitoring". In: *Proceedings of the 6th International Conference on Mobile Systems, Applications, and Services*. MobiSys '08. Breckenridge, CO, USA: ACM, 2008, pp. 29–39. ISBN: 978-1-60558-139-2. DOI: 10.1145/1378600.1378605. URL: <http://doi.acm.org/10.1145/1378600.1378605>.
- [12] Hossein Falaki, Ratul Mahajan, and Deborah Estrin. "SystemSens: A Tool for Monitoring Usage in Smartphone Research Deployments". In: *Proceedings of the Sixth International Workshop on MobiArch*. MobiArch '11. Bethesda, Maryland, USA: ACM, 2011, pp. 25–30. ISBN: 978-1-4503-0740-6. DOI: 10.1145/1999916.1999923. URL: <http://doi.acm.org/10.1145/1999916.1999923>.
- [13] K. Farkas and I. Lendák. "Simulation environment for investigating crowd-sensing based urban parking". In: *2015 International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*. June 2015, pp. 320–327. DOI: 10.1109/MTITS.2015.7223274.
- [14] Jesus Favela et al. "Activity Recognition for Context-aware Hospital Applications: Issues and Opportunities for the Deployment of Pervasive Networks". In: *Mob. Netw. Appl.* 12.2-3 (Mar. 2007), pp. 155–171. ISSN: 1383-469X. DOI: 10.1007/s11036-007-0013-5. URL: <http://dx.doi.org/10.1007/s11036-007-0013-5>.
- [15] C. Fiandrino et al. "CrowdSenSim: a Simulation Platform for Mobile Crowd-sensing in Realistic Urban Environments". In: *IEEE Access* 5 (Feb. 2017), pp. 3490–3503. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2017.2671678.
- [16] C. Fiandrino et al. "Sociability-Driven Framework for Data Acquisition in Mobile Crowdsensing Over Fog Computing Platforms for Smart Cities". In: *IEEE Transactions on Sustainable Computing* 2.4 (Oct. 2017), pp. 345–358. ISSN: 2377-3782. DOI: 10.1109/TSUSC.2017.2702060.
- [17] C. Fiandrino et al. "Sociability-Driven User Recruitment in Mobile Crowd-sensing Internet of Things Platforms". In: *2016 IEEE Global Communications Conference (GLOBECOM)*. Dec. 2016, pp. 1–6. DOI: 10.1109/GLOCOM.2016.7842272.
- [18] X. Gong et al. "Exploiting Social Trust Assisted Reciprocity (STAR) Toward Utility-Optimal Socially-Aware Crowdsensing". In: *IEEE Transactions on Signal and Information Processing over Networks* 1.3 (Sept. 2015), pp. 195–208. ISSN: 2373-776X. DOI: 10.1109/TSIPN.2015.2470110.
- [21] Nancy B Grimm et al. "Global change and the ecology of cities". In: *science* 319.5864 (2008), pp. 756–760. DOI: 10.1126/science.1150195.

- [22] N. Haderer, R. Rouvoy, and L. Seinturier. "A preliminary investigation of user incentives to leverage crowdsensing activities". In: *IEEE International Conference on Pervasive Computing and Communications Workshops*. Mar. 2013, pp. 199–204. doi: 10.1109/PerComW.2013.6529481.
- [23] K. Han, C. Zhang, and J. Luo. "Taming the Uncertainty: Budget Limited Robust Crowdsensing Through Online Learning". In: *IEEE/ACM Transactions on Networking* 24.3 (June 2016), pp. 1462–1475. issn: 1063-6692. doi: 10.1109/TNET.2015.2418191.
- [24] K. Han, C. Zhang, and J. Luo. "Taming the Uncertainty: Budget Limited Robust Crowdsensing Through Online Learning". In: *IEEE/ACM Transactions on Networking* 24.3 (June 2016), pp. 1462–1475. issn: 1063-6692. doi: 10.1109/TNET.2015.2418191.
- [25] Y. He, Y. Li, and S. D. Bao. "Fall detection by built-in tri-accelerometer of smartphone". In: *Proceedings of 2012 IEEE-EMBS International Conference on Biomedical and Health Informatics*. Jan. 2012, pp. 184–187. doi: 10.1109/BHI.2012.6211540.
- [26] H. Huang et al. "A Truthful Double Auction Mechanism for Crowdsensing Systems with Max-Min Fairness". In: *2017 IEEE Wireless Communications and Networking Conference (WCNC)*. Mar. 2017, pp. 1–6. doi: 10.1109/WCNC.2017.7925505.
- [27] Kuan Lun Huang, Salil S. Kanhere, and Wen Hu. "Are You Contributing Trustworthy Data?: The Case for a Reputation System in Participatory Sensing". In: *Proceedings of the 13th ACM International Conference on Modeling, Analysis, and Simulation of Wireless and Mobile Systems*. MSWIM '10. Bodrum, Turkey: ACM, 2010, pp. 14–22. isbn: 978-1-4503-0274-6. doi: 10.1145/1868521.1868526. url: <http://doi.acm.org/10.1145/1868521.1868526>.
- [28] Panagiotis G. Ipeirotis, Foster Provost, and Jing Wang. "Quality Management on Amazon Mechanical Turk". In: *Proceedings of the ACM SIGKDD Workshop on Human Computation*. HCOMP '10. Washington DC: ACM, 2010, pp. 64–67. isbn: 978-1-4503-0222-7. doi: 10.1145/1837885.1837906. url: <http://doi.acm.org/10.1145/1837885.1837906>.
- [29] Raj Jain, Dah-Ming Chiu, and W. Hawe. "A Quantitative Measure Of Fairness And Discrimination For Resource Allocation In Shared Computer Systems". In: *CoRR* cs.NI/9809099 (1998).
- [30] P. P. Jayaraman et al. "Scalable Energy-Efficient Distributed Data Analytics for Crowdsensing Applications in Mobile Environments". In: *IEEE Transactions on Computational Social Systems* 2.3 (Sept. 2015), pp. 109–123. issn: 2329-924X.

- [31] F. Kalim, J. P. Jeong, and M. U. Ilyas. "CRATER: A Crowd Sensing Application to Estimate Road Conditions". In: *IEEE Access* 4 (2016), pp. 8317–8326. doi: 10.1109/ACCESS.2016.2607719.
- [32] K. Kapetanakis and S. Panagiotakis. "Efficient Energy Consumption's Measurement on Android Devices". In: *2012 16th Panhellenic Conference on Informatics*. Oct. 2012, pp. 351–356. doi: 10.1109/PCi.2012.29.
- [33] M. Karaliopoulos, O. Telelis, and I. Koutsopoulos. "User recruitment for mobile crowdsensing over opportunistic networks". In: *2015 IEEE Conference on Computer Communications (INFOCOM)*. Apr. 2015, pp. 2254–2262. doi: 10.1109/INFOCOM.2015.7218612.
- [34] W.Z. Khan et al. "Mobile Phone Sensing Systems: A Survey". In: *IEEE Communications Surveys Tutorials* 15.1 (First Quarter 2013), pp. 402–427. ISSN: 1553-877X. doi: 10.1109/SURV.2012.031412.00077.
- [35] Sunyoung Kim et al. "Creek Watch: Pairing Usefulness and Usability for Successful Citizen Science". In: *Proc. of the SIGCHI Conference on Human Factors in Computing Systems*. CHI. Vancouver, BC, Canada: ACM, 2011, pp. 2125–2134. ISBN: 978-1-4503-0228-9. doi: 10.1145/1978942.1979251.
- [36] Nicholas D. Lane et al. "Piggyback CrowdSensing (PCS): Energy Efficient Crowdsourcing of Mobile Sensor Data by Exploiting Smartphone App Opportunities". In: *11th ACM Conference on Embedded Networked Sensor Systems*. SenSys. Roma, Italy, 2013, pp. 1–14. ISBN: 978-1-4503-2027-6. doi: 10.1145/2517351.2517372.
- [37] Robert LiKamWa et al. "Draining Our Glass: An Energy and Heat Characterization of Google Glass". In: *Proc. of 5th Asia-Pacific Workshop on Systems*. APSys. Beijing, China: ACM, 2014, pp. 1–7. ISBN: 978-1-4503-3024-4. doi: 10.1145/2637166.2637230.
- [38] C.H. Liu et al. "Energy-Aware Participant Selection for Smartphone-Enabled Mobile Crowd Sensing". In: *IEEE Systems Journal* PP.99 (2015), pp. 1–12. ISSN: 1932-8184. doi: 10.1109/JSYST.2015.2430362.
- [39] Y. Liu et al. "GreenDroid: Automated Diagnosis of Energy Inefficiency for Smartphone Applications". In: *IEEE Transactions on Software Engineering* 40.9 (Sept. 2014), pp. 911–940. ISSN: 0098-5589. doi: 10.1109/TSE.2014.2323982.
- [40] Jeffrey W. Lockhart, Tony Pulickal, and Gary M. Weiss. "Applications of Mobile Activity Recognition". In: *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*. UbiComp '12. Pittsburgh, Pennsylvania: ACM, 2012, pp. 1054–1058. ISBN: 978-1-4503-1224-0. doi: 10.1145/2370216.2370441. URL: <http://doi.acm.org/10.1145/2370216.2370441>.

- [41] Vitali Loseu et al. "Applications of Sensing Platforms with Wearable Computers". In: *Proceedings of the 3rd International Conference on Pervasive Technologies Related to Assistive Environments*. PETRA '10. Samos, Greece: ACM, 2010, 53:1–53:5. ISBN: 978-1-4503-0071-1. DOI: 10.1145/1839294.1839358. URL: <http://doi.acm.org/10.1145/1839294.1839358>.
- [42] M. Marjanović, S. Grubeša, and I. P. Žarko. "Air and noise pollution monitoring in the city of Zagreb by using mobile crowdsensing". In: *2017 25th International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*. Sept. 2017, pp. 1–5. DOI: 10.23919/SOFTCOM.2017.8115502.
- [43] Emiliano Miluzzo et al. "CenceMe: Injecting Sensing Presence into Social Networking Applications". In: *Proceedings of the 2Nd European Conference on Smart Sensing and Context*. EuroSSC'07. Kendal, England: Springer-Verlag, 2007, pp. 1–28. ISBN: 3-540-75695-7, 978-3-540-75695-8. URL: <http://dl.acm.org/citation.cfm?id=1775377.1775379>.
- [45] Federico Montori, Luca Bedogni, and Luciano Bononi. "Distributed Data Collection Control in Opportunistic Mobile Crowdsensing". In: *Proc. of the 3rd Workshop on Experiences with the Design and Implementation of Smart Objects*. SMARTOBJECTS. Snowbird, Utah, USA: ACM, 2017, pp. 19–24. ISBN: 978-1-4503-5141-6. DOI: 10.1145/3127502.3127509.
- [46] R. Mulero et al. "An AAL system based on IoT technologies and linked open data for elderly monitoring in smart cities". In: *2nd International Multidisciplinary Conference on Computer and Energy Science (SpliTech)*. July 2017, pp. 1–6.
- [47] J. Ni et al. "Security, Privacy, and Fairness in Fog-Based Vehicular Crowdsensing". In: *IEEE Communications Magazine* 55.6 (2017), pp. 146–152. ISSN: 0163-6804. DOI: 10.1109/MCOM.2017.1600679.
- [48] Bei Pan et al. "Crowd Sensing of Traffic Anomalies Based on Human Mobility and Social Media". In: *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. SIGSPATIAL'13. Orlando, Florida: ACM, 2013, pp. 344–353. ISBN: 978-1-4503-2521-9. DOI: 10.1145/2525314.2525343. URL: <http://doi.acm.org/10.1145/2525314.2525343>.
- [49] D. Peng, F. Wu, and G. Chen. "Data Quality Guided Incentive Mechanism Design for Crowdsensing". In: *IEEE Transactions on Mobile Computing* 17.2 (Feb. 2018), pp. 307–319. ISSN: 1536-1233. DOI: 10.1109/TMC.2017.2714668.
- [50] Jia Peng et al. "Fair Energy-Efficient Sensing Task Allocation in Participatory Sensing with Smartphones". In: *The Computer Journal* 60.6 (2017), pp. 850–865.

- [51] G. P. Perrucci, F. H. P. Fitzek, and J. Widmer. "Survey on Energy Consumption Entities on the Smartphone Platform". In: (May 2011), pp. 1–6. ISSN: 1550-2252. doi: 10.1109/VETECS.2011.5956528.
- [52] R. Pryss et al. "Mobile Crowd Sensing Services for Tinnitus Assessment, Therapy, and Research". In: *2015 IEEE International Conference on Mobile Services*. June 2015, pp. 352–359. doi: 10.1109/MobServ.2015.55.
- [53] D. Puiu et al. "CityPulse: Large Scale Data Analytics Framework for Smart Cities". In: *IEEE Access* 4 (2016), pp. 1086–1108. ISSN: 2169-3536. doi: 10.1109/ACCESS.2016.2541999.
- [54] Rajib Kumar Rana et al. "Ear-phone: An End-to-end Participatory Urban Noise Mapping System". In: *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*. IPSN '10. Stockholm, Sweden: ACM, 2010, pp. 105–116. ISBN: 978-1-60558-988-6. doi: 10.1145/1791212.1791226. URL: <http://doi.acm.org/10.1145/1791212.1791226>.
- [55] Sasank Reddy and Vids Samanta. *Urban Sensing: Garbage Watch*. UCLA Center for Embedded Networked Sensing. 2011.
- [56] Sasank Reddy et al. "Examining Micro-payments for Participatory Sensing Data Collections". In: *Proceedings of the 12th ACM International Conference on Ubiquitous Computing*. UbiComp '10. Copenhagen, Denmark: ACM, 2010, pp. 33–36. ISBN: 978-1-60558-843-8. doi: 10.1145/1864349.1864355. URL: <http://doi.acm.org/10.1145/1864349.1864355>.
- [58] Pablo Serrano et al. "Per-frame energy consumption in 802.11 devices and its implication on modeling and design". In: *IEEE/ACM Transactions on Networking* 23.4 (2015), pp. 1243–1256.
- [59] W. Sherchan et al. "Using On-the-Move Mining for Mobile Crowdsensing". In: *IEEE 13th International Conference on Mobile Data Management*. July 2012, pp. 115–124.
- [60] V. Sivaraman et al. "HazeWatch: A participatory sensor system for monitoring air pollution in Sydney". In: *IEEE Conference on Local Computer Networks (LCN) - Workshops*. Oct. 2013, pp. 56–64. doi: 10.1109/LCNW.2013.6758498.
- [62] J. W. Sung and S. J. Han. "Data bundling for energy efficient communication of wearable devices". In: *IEEE 40th Conference on Local Computer Networks (LCN)*. Oct. 2015, pp. 321–328. doi: 10.1109/LCN.2015.7366326.
- [63] Kun Tan et al. "Fine-grained Channel Access in Wireless LAN". In: *Proc. of the ACM SIGCOMM*. New Delhi, India: ACM, 2010, pp. 147–158. ISBN: 978-1-4503-0201-2. doi: 10.1145/1851182.1851202.

- [64] Jing Wang et al. "Towards energy-efficient task scheduling on smartphones in mobile crowd sensing systems". In: *Computer Networks* 115 (2017), pp. 100–109. ISSN: 1389-1286. DOI: <https://doi.org/10.1016/j.comnet.2016.11.020>.
- [65] L. Wang et al. "effSense: A Novel Mobile Crowd-Sensing Framework for Energy-Efficient and Cost-Effective Data Uploading". In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 45.12 (Dec. 2015), pp. 1549–1563. ISSN: 2168-2216. DOI: [10.1109/TSMC.2015.2418283](https://doi.org/10.1109/TSMC.2015.2418283).
- [67] D. Willkomm et al. "Primary user behavior in cellular networks and implications for dynamic spectrum access". In: *IEEE Communications Magazine* 47.3 (Mar. 2009), pp. 88–95. ISSN: 0163-6804. DOI: [10.1109/MCOM.2009.4804392](https://doi.org/10.1109/MCOM.2009.4804392).
- [68] W. Wu et al. "Energy-Efficient Transmission With Data Sharing in Participatory Sensing Systems". In: *IEEE Journal on Selected Areas in Communications* 34.12 (Dec. 2016), pp. 4048–4062. ISSN: 0733-8716. DOI: [10.1109/JSAC.2016.2621359](https://doi.org/10.1109/JSAC.2016.2621359).
- [69] Haoyi Xiong et al. "EEMC: Enabling Energy-Efficient Mobile Crowdsensing with Anonymous Participants". In: *ACM Trans. Intell. Syst. Technol.* 6.3 (Apr. 2015), 39:1–39:26. ISSN: 2157-6904. DOI: [10.1145/2644827](https://doi.org/10.1145/2644827). URL: <http://doi.acm.org/10.1145/2644827>.
- [70] H. Xiong et al. "EMC3: Energy-efficient data transfer in mobile crowdsensing under full coverage constraint". In: *IEEE Transactions on Mobile Computing* 14.7 (July 2015), pp. 1355–1368. ISSN: 1536-1233. DOI: [10.1109/TMC.2014.2357791](https://doi.org/10.1109/TMC.2014.2357791).
- [71] D. Yang et al. "Incentive Mechanisms for Crowdsensing: Crowdsourcing With Smartphones". In: *IEEE/ACM Transactions on Networking* 24.3 (June 2016), pp. 1732–1744. ISSN: 1063-6692. DOI: [10.1109/TNET.2015.2421897](https://doi.org/10.1109/TNET.2015.2421897).
- [72] A. Zanella et al. "Internet of Things for Smart Cities". In: *IEEE Internet of Things Journal* 1.1 (Feb. 2014), pp. 22–32. ISSN: 2327-4662. DOI: [10.1109/JIOT.2014.2306328](https://doi.org/10.1109/JIOT.2014.2306328).
- [73] Y. Zhang and M. van der Schaar. "Reputation-based incentive protocols in crowdsourcing applications". In: *2012 Proceedings IEEE INFOCOM*. Mar. 2012, pp. 2140–2148. DOI: [10.1109/INFCOM.2012.6195597](https://doi.org/10.1109/INFCOM.2012.6195597).
- [74] Q. Zhao et al. "Fair energy-efficient sensing task allocation in participatory sensing with smartphones". In: *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*. Apr. 2014, pp. 1366–1374. DOI: [10.1109/INFOCOM.2014.6848070](https://doi.org/10.1109/INFOCOM.2014.6848070).





## Online resources

- [1] Android Developers. *Dashboard*. Accessed February 20, 2018. 2018. URL: <https://developer.android.com/about/dashboards/index.html>.
- [2] Android Developers. *Processes and Application Lifecycle*. Accessed March 07, 2018. URL: <https://developer.android.com/guide/components/activities/process-lifecycle.html>.
- [19] Google APIs for Android. *Location Request*. Accessed March 07, 2018. URL: <https://developers.google.com/android/reference/com/google/android/gms/location/LocationRequest>.
- [20] Google Source. *Sensor Stack*. Accessed January 23, 2018. 2017. URL: <https://source.android.com/devices/sensors/sensor-stack>.
- [44] Monsoon Solutions Inc. *HIGH VOLTAGE POWER MONITOR*. Accessed March 08, 2018. URL: <https://www.msoon.com/online-store>.
- [57] RestApiTutorial.com. *What Is REST?* Accessed March 05, 2018. URL: <http://www.restapitutorial.com/lessons/whatisrest.html#>.
- [61] Statcounter. *Mobile Operating System Market Share Worldwide*. Accessed January 23, 2018. URL: <http://gs.statcounter.com/os-market-share/mobile/worldwide>.
- [66] Wiko. *Wiko Sunny*. Accessed March 08, 2018. URL: <http://it.wikomobile.com/m1330-sunny>.