POLITECNICO DI TORINO

Master Degree in Computer Engineering

Master Thesis

Modeling Mobility, Task Allocation and Data Collection for Mobile Crowdsensing Simulators

Advisor:
Prof. Paolo Giaccone

Candidate:
Piergiorgio Vitello

External Supervisors
Dr. Claudio Fiandrino
Andrea Capponi

Aprile 2018

Summary

Mobile Crowd Sensing (MCS) has recently gained popularity becoming an appealing paradigm for sensing and collecting data. MCS is sensing paradigm that leverages active participation of citizens to improve existing infrastructures without the need of further investments. Thanks to the exponential diffusion of smartphones, MCS is considered an emerging area of interest for researchers. The key fact is that mobile devices have not only computing and communication resources, but they also offer the possibility to exploit a rich set of sensors for enabling new applications across a large variety of domains. In MCS systems users typically contribute data individually, forming groups in which users collaborate to contribute data is an efficient solution. The key idea for each group is to exchange data between peers and elect an owner, who is the only responsible for data delivery. To this end, it is crucial to propose policies to form groups and elect a responsible according to the requirements of the campaign (e.g., the technology employed). From an energy perspective, it is well known that users sustain costs mostly due to reporting than sensing operations. Hence, leveraging device to device (D2D) communications within a group of citizens that sense the same phenomena in the same area is a win-win strategy in specific domains of interest (e.g., mapping air pollution). In this work, I propose and compare three different policies to form groups and perform owner election, analyzing how to allocate tasks to a group of peers instead of an individual. In order to evaluate the proposed framework, this thesis focused in the improvement of CrowdSenSim a simulator for MCS systems. The simulator is a discrete-event simulator in which the participants contribute data to the MCS system. The renovation of the simulator allows the participants to move in a real city environment of any place

on Earth. The simulator is scalable thanks to a novel pedestrian mobility with high level precision. I proposed a procedure to augment the precision (AOP) of the graph describing the street network provided by OpenStreetMap. Moreover, i implemented in the simulator two different mobility models one synthetic and one based on a realistic dataset.

Acknowledgements

Molte sono le persone che vorrei ringraziare giunto al termine di questo mio percorso. In primis il mio relatore il Professor Paolo Giaccone, per avermi dato la possibilità di effettuare il periodo di ricerca tesi a Lussemburgo e per avermi dimostrato la sua disponibilità anche a distanza.

Gli ultimi mesi del mio percorso di studi, trascorsi in Lussemburgo, sono stati per me bellissimi e mi hanno aiutato a crescere, soprattutto a livello lavorativo. Per questo, vorrei ringraziare Andrea e Claudio per avermi seguito costantemente e per la disponibilità mostrata in ogni momento, non solo come miei supervisori, ma anche a livello umano. Questa esperienza mi ha fatto conoscere molte persone che vorrei ringraziare per aver reso questa esperienza speciale, porterò sempre con me un bel ricordo di ognuno di voi. Tra tutti, in particolar modo, vorrei ringraziare Federico, Enrico, Sinan, Koby, Mattia, Ilaria e Bayar.

Il ringraziamento più grande va alla mia grande famiglia, che mi è stata sempre vicina in ogni momento. I miei genitori, per il loro affetto e per avermi sempre supportato nelle mie scelte, le mie sorelle Emanuela, Federica e Elisabetta senza le quali non sarei diventato l'uomo che sono, la nonna Fofò per l'amore che mi ha trasmesso. Non posso non ringraziare la famiglia orbasanese, per avermi sempre aiutato in questi anni di "trasferta", specialmente il mio fratellone Luca. Un grazie speciale a Claudia, per l'affetto e l'appoggio che mi ha dato, soprattutto nei momenti più difficili. Gli anni vissuti a Torino non sarebbero stati gli stessi senza i miei quattro compagni di viaggio Richard, Lorenzo, Alessio e Silvio, li ringrazio con la speranza di poter vivere insieme altre avventure future. Infine, ma non per questo meno importanti, vorrei ringraziare Carlotta, Ciccio, Sara, Roberta e Damiano. Grazie per avermi ,ognuno in maniera diversa, supportarto e sopportarto sempre.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

1.1 Motivation

The breathtaking evolution of mobile devices has constantly increased in the last years.Nowadays they are an essential part of people's daily life embedding computational and communication capabilities. This exponential diffusion has promoted the proliferation of built-in sensors, making them a huge source of data available for the application of the Internet of Things (IoT) paradigm, a key component for making real the smart city vision [71, 53]. With the aim of improving citizens' quality of life, significant research efforts are undergoing to provide citizens innovative and sustainable solutions for public services such as environmental monitoring, social network analysis, health-care, intelligent transportation systems, and public safety. To illustrate with a few examples, HazeWatch[62] relies on active citizen participation to monitor air pollution and is currently employed by the National Environment Agency of Singapore on a daily basis. Creekwatch[40] is an application for smartphones developed by the IBM Almaden research center. It allows the monitoring of the conditions of watershed through crowdsensed collected data about the amount of water in the river bed, the amount of trash in the river bank, the flow rate, and a picture of the waterway. Garbage Watch[55] and WasteApp[7] employ citizens to monitor the content of recycling bins with the objective of improving the recycling program. Despite their growth,all these applications are facing different barriers. Some of

them are technological, resulting from the heterogeneity of hardware/software platforms,causing a raise of costs for development of MCS. A large participation of users is fundamental for the success of a sensing campaign. User contributing data sustain costs, e.g., energy consumption or cellular data plan and it is essential to reward them fro joining the sensing campaign. In MCS systems users typically contribute data individually, The key idea of this thesis is that there is no need of each mobile device connected to WiFi or cellular interfaces. The presence of a local authority avoid to send several times similar information, which represents a data overhead to process (e.g., many samples of temperature from users in the place at the same time). The main goal of this work is to form groups of contributing users, in order to exchange data between them and elect an owner, who is the only responsible for data delivery. The aim is to propose an efficient data collection and the possibility to perform group-based task assignment instead on individual-basis. To assess the performance of MCS systems requires the contribution from a large number of participants. However, the development of large scale testbeds is often not feasible. An essential solution is using simulations. However, the number of existing simulation tools appropriate for inspecting MCS efforts is limited. CrowdSenSim[23] is the first simulator for MCS systems, designed exclusively for MCS purposes. It provide the researchers a tool capable of performing large scale simulations over realistic urban environments. Unfortunately, the actual version of CrowdSenSim presents different weak points, the most significant is the absence of scalability in the mobility module. Performing simulations in complex environments, such as modern cities, requires the simulation platform to be scalable while providing at the same time precise and detailed information. In order to fill these gaps I renewed CrowdSenSim changing many modules and adding new features.

1.2 Contribution

This thesis investigates the feasibility of a novel solution for a MCS Framework and analyzes advantages brought by it. This new model that I developed is based on a collaborative framework (CF). The typical MCS architecture is composed by a

data collector and a set of participants . Typically MCS systems perform sensing on individual basis, each user sends his amount of data collected during sensing. Moreover the task allocation is managed by the DC, which assigns a task to a single participant. The new architecture aims to enhance the user experience in terms of energy efficiency and fairness between users. To reach such results with the new CF I introduce the concept of group of users in MCS, these groups will have multiple functions, first create efficient data collection through device to device (D2D) communication and second give the possibility to data collector to manage group-based task assignment instead on individual-basis. The idea behind the collaborative sensing is that the users will not deliver their data directly to the DC, but they will report it to an elected user through D2D connections, the elected user will assume the role of group owner (GO) and will act as relay for other peers in the same group. In this work I decided to exploit WiFi direct, formally known as WiFi Peer to Peer. It is a communication between Wi-Fi devices which enables to directly connect each other without connecting to a legacy access point. To develop CF is required a cluster algorithm through which we will group different devices. For each cluster a chosen node will take the role of GO, this choice has to be taken according to different factors, in order to find the proper policy that will give the best results. In this thesis i propose three different grouping strategies: static, POI based , dynamic. Each one is composed by a cluster algorithm and a group owner election, Furthermore to every strategy corresponds a different task allocation need. For instance, the static approach can be applied for task in which the main purpose is to cover all the territory with an equal density, the dynamic is useful for tasks that aim to maximize the contact-time between the group, while the POI can be used in case of task concerning a particular type of places inside a city. Extensive simulation results are provided to demonstrate the advantages of our proposed scheme. The Simulator used is CrowdSenSim, developed at the University of Luxembourg. During my internship at UniLu, I became involved in the elaboration of new releases of CrowdSenSim. The main contribution that I gave to the simulator is in the mobility domain, i designed a novel pedestrian mobility with high level precision for use in crowdsensing smart city simulators. I proposed a procedure to augment the precision (AOP) of the graph describing

the street network provided by OpenStreetMap. Second, i implemented in the simulator two different mobility models one synthetic and one based on a realistic dataset. Through performance evaluation conducted with CrowdSenSim, I verified the effectiveness of the adopted AOP approach and then evaluate the accuracy obtained by the two arrival models by implementing metrics commonly employed for pedestrian mobility such as the distribution of the contact rate and of the contact stability. Moreover, I developed for CrowdSenSim other new features not related with the mobility module, such as the profiling of energy consumption based on realistic smartphone measurements and the implementation of different data reporting mechanism and data collection frameworks.

1.3 Organization

The thesis is organized as follows:

- Chapter 2 starts describing the current state of the art concerning Mobile Crowdsensing and its simulators, analyzing one of them in particular, CrowdSenSim. Then it introduces the landscape of what has be done until now on reporting strategies and task allocation, discussing the current problems and limitations.
- Chapter 3 illustrates the new version of CrowdSenSim, describing the new implemented mobility and other improvements.
- Chapter 4 proposes the collaborative framework for MCS, it is divided in three sections. The first presents the methodology to create the groups of users and the different way of clustering. The second section introduces the task allocation methods algorithm. At the end the third evaluates the framework with the results obtained through CrowdSenSim.
- Chapter 5 concludes the work and presents future research directions on the topic.

Chapter 2

Background on Mobile Crowdsensing

2.1 State of the art

In this chapter I introduce the main aspects of mobile crowdsensing, at first I present a general background of MCS, in the second part I give an idea of the simulation tools available for MCS, at the end of the chapter I focus my research on explaining the fundamental characteristics of a specific simulator, CrowdSenSim.

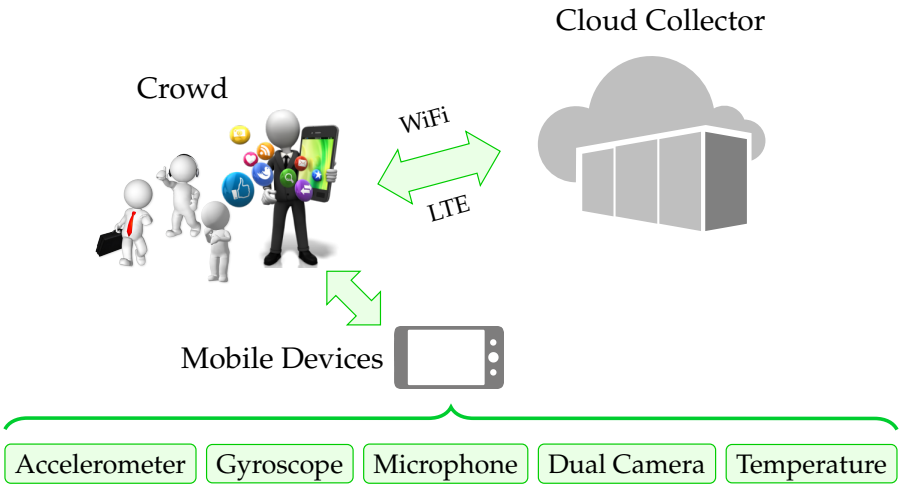


Figure 2.1. Cloud-based MCS system [24]

The global market of Internet of Things (IoT) reached 598.2 Billion USD in 2015 and the previsions talk about an increase that will reach 724.2 Billion USD by 2023. The expected annual growth rate of the market is 13.2% until 2023. [35]

Modern IoT devices, as smartphones or wearables, constantly communicate with each other and with users, to continuously enable computing power. They are equipped with a set of sensors and have communication and computing capabilities. They own several sensors capable of perceiving the surrounding environment. The growing availability of different types of sensors on mobile devices makes them suitable for the development of the so called smart cities. It is difficult to find a homogeneous definition of smart city, as the concept of "smartness" regards different fields: environment, quality of life, planning and technology.

Smart cities aim to improve to improve the citizens quality of life. For instance, it is possible to simplify the citizen daily routines, for example improving public services or by developing novel solutions for healthcare and public safety. Moreover, Smart cities have the ambition to promote the values of sustainable development e rational use of resources in our metropolis. The IoT paradigm is the fundamental candidate building block to develop sustainable ICT platforms for Smart Cities. Mobile CrowdSensing (MCS) is an emerging paradigm that exploits the diffusion of mobile devices in order to collect data in an efficient way. In its applications, the key element is the human involvement: MCS employs the devices' sensors, daily carried by users, for larger scale studies. MCS is the most natural form of co-operation between people living smart cities, It allow users to contribute in complex problem solving. In this scenario, the network of 'sensors' consists of people, the group of human users are called 'crowd'. Users can exploit their modern smartphones and devices to collect and share several types of data [27] The users devices have a large multi-sensing capabilities including gps localization, movement, light, and audio and visual sensors. They become citizen sensors by performing some form of sensing [28].

The human involvement might have positive and negative aspects. On the positive side, MCS makes it easier to develop a low cost coverage network in the area of interest and users take care of the nodes of this network, which are their

mobile devices. The downside is that there could be privacy issues and problems related to the fact that the number of devices, the availability of sensors and the quality of data are subject to unpredictable changes over time. There are two types of crowdsensing [25] [49]:

- Participatory sensing
- Opportunistic sensing

In the first, the users decide what kind of data and when to collect it and report it. An example is an user taking individual readings and reporting them to the data collector. An example of it is the Participatory Urbanism project [48] which recruits urban people in order to perform a collection of air quality data sensors using mobile devices. Opportunistic sensing: data collection is performed on the background, without the active involvement of users by do not let them know about the active execution of sensing application. In opportunistic sensing the user may not even be aware that he is actually participating. An example is BubbleSensing [44] where they introduce an opportunistic protocol which collects locaton data from mobile devices. In [17] Chen et al. Developed a distributed algorithm for maximizing the utility of sensing data collection of smartphone under a budget cost costraint. It is inspired by stochastic network optimization and distributed scheduling.

The opportunistic sensing has to deal with some issues in terms of battery usage, the user does not have a consciousness of his contribution to the campaign and consequentially does not have the perception of the actual energy consumption due to the collection of data in the background. On the other side, In the participatory reveals the opposite problem. Users can correlate the battery drain of their devices to the participation to the MCS campaign and as consequence their willingness to contribute could decrease dramatically.

The management of a MCS process is a demanding task, as it involves a number of technical issues, such as: the choice of region to monitor, the quantification of the necessary sensors' density to obtain accuracy, the valuation of a good balance between accuracy and exploitation of resources. Moreover, social issues arise as well: the choice of incentives for users' active participations and the trust of people

involved. As Fig. 2.1 shows in MCS data collection architectures the participants contribute information from mobile devices' sensors. This information is then delivered to a collector, typically located in the cloud. The collector receives the data and proceed on processing and analysis it. The users deliver collected data using cellular 3G/LTE/4G or WLAN interfaces. Therefore, it is essential to analyze and assess the costs of sensing and data reporting for users, while maximizing the utility of the collected information. Crowdsensing plays an meaningful role in numerous application domains such as health care, collecting cartographic, environment monitoring, urban sensing, transportation, social networks and safety. To illustrate with few examples, AndWellness (Hicks et al., 2010) developed a personal data collection system, where users through mobile devices can record their individual habits . Thie system may have several outcomes, could be employed to assess HIV+ patients monitoring their daily behavior. Another system is Biketastic [56], which purpose is to make available to cyclists information about different bike routes, in order to facilitate them for choosing right path. The goal is obtained by collecting device sensors bikers during route experience. In addition to the GPS It used accelerometer to sense noise level and roughness. An example of participatory sensing for air monitoring is available at GasMobile [32], while NoiseMap [59] bring us an application to map the noise pollution of a city. A fascinating application is SenseMyCity [58]. The platform collects data and information from sensors equipped in phones of vehicles drivers in the city of Porto. Afterwards, they combined the acquired data to show the map of the fuel consumption in the city and the bus drivers stress by using cardiac sensors.

MCS domain has to challenge with many issues, such as handling big data, fusion of data sensors, privacy of users, or the development of multi platform algorithms.

2.2 Task allocation

In *participatory* MCS, every user is engaged to collect data. The MCS system assigns a particular task to an participants available, in order to accomplish it. As mentioned before MCS can be seen as a problem-solving distributed system, where

a complex problem is divided above all the user of the campaign. This solution requires a "chief brain" able to manage all the tasks and all the participants.

One of the most important issues regarding the MCS is the efficient allocation of sensing tasks.

To execute a task in MCS typically, many parameters has to be defined, such as:

- the purpose of the task
- the type of effort that the specified task requires to the participant.
- the presence or not of user incentives and rewarding to compensate the participants for their contribution.
- a level of required quality for sensing data.
- A given location (expressed as geographical coordinates) and a given ray, defining a task area.
- The total amount of data collected to accomplish the task .

Most of the work about task allocation for MCS usually aim to maximize a set of parameters, e.g., the amount/temporal and or spatial coverage of the contributed data or the QoI while minimizing at the same time the costs, such as energy consumption or monetary rewards [75].

Reddy et al. in [54] try to recruit a chosen number of participants in order to maximize the spatial coverage. Other studies tried to extend this type of coverage-maximization allocation, focusing their effort to give importance to participants, aiming attention at incentive mechanism [37] or travel time budget [33]. Meanwhile, work such as [61][1][30] attempts to minimize the incentive budget and/or energy consumption under a full or high probabilistic spatio and/or temporal coverage constraint.

Wu et al. [69] 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. [67] 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. [31] 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. [43] 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.

2.3 Simulators for MCS

The main issue during the development of novel MCS application is the cost to handle facing these large scale studies. Therefore becomes fundamental to perform extensive analysis domain and risk assessments before start to develop a novel MCS system.

Following this purpose, the first essential step is using simulations [60] [50] [52]. The creation of simplified representation of the domain of interest is an essential solution for testing MCS.

Simulating MCS scenarios is useful to analyze the possible behaviours of the crowd and assess if the system developed provide the required level of service quality.

However, the number of existing simulation tools appropriate for inspecting MCS efforts is limited. I will describe the main properties of each tool in the following.

Network Simulator 3 (NS-3) is discrete-event network simulator, designed for networking research and education. Tanas et al. presented [64] a cowsensing simulation study basde on NS-3. The goal is to estimate the performance of a crowdsensing network considering that the wireless interface in ad-hoc network mode together with mobility properties of the nodes.

NS-3 is focused on ensure highly accurate estimations of network behavior. As consequence of having such high accuracy on network properties, the simulation has to face with a loose of scalability.

For this reason with NS-3, the emulation of tens of thousands of devices is not feasible. The purpose is to capture specific network attitudes, such as the changes of the TCP congestion window. Therefore the timing of a single simulation is not in the order of hours or days but typically in minutes . Indeed, NS-3 is not able to simulate the duration of real sensing campaigns.

In [22] the authors introduce instead a simulation environment adapted to investigate performance of crowdsensing applications in an urban parking scenario. The possible applications are only related to the parking domain, but the authors assert that the simulation environment can be enlarged to other scenarios.

However, only drivers are take in consideration as type of contributing users. Farkas and Lendàk consider humans as participants only if they are travelling between two parking spots. In order to be applicable in a large scale environment, a crowdsensing simulator has to take into account data coming from IoT devices' carried by human individuals, in any situation.

Mehdi et al. propose CupCarbon [45], a discrete-event wireless sensor network (WSN) simulator for IoT and smart cities. It exploit the features of OpenStreetMap in order to simulate WSN on realistic urban environments. The simulator is not feasible for scenarios with thousands of users, who wants to start a simulation has to deploy each node and sensors on the map.

2.4 CrowdSenSim

CrowdSenSim, is a discrete-event simulator for mobile crowdsensing developed at University of Luxembourg. It enables simulations of crowdsensing activities in large-scale urban scenarios. CrowdSenSim can be used to develop novel solutions in data collection, task assignment, monitoring and resource management. For instance this tool allows researchers and scientist to test and evaluate the goodness and the feasibility of a crowdsensing application, trough the simulation of real smart cities, where are walking a specified number of users involved on different MCS campaigns. During my internship at University of Luxembourg, i worked on a new release of the simulator, CrowdSenSim 1.1.0 (CS1). It is available for download at [20], where can be found both the source code and a running environment built on an Ubuntu virtual machine. This section introduces the features and the characteristics of the old version of CrowdSenSim (CS0). The code of the simulator is implemented entirely in *C++* and is composed by independent modules.

2.4.1 Architecture

Fig. 2.2 shows the structure of CS0, it is immediately apparent the division in independents modules, this type of architecture was made to give a high level of

scalability and allow future developers to implement and modify some modules while keeping untouched others and preserving the main structure. This was the aspect that has led the develop of my work on CrowdSenSim, in which i completely renewed some modules which will be described in 3.2.

City Layout Module

In order to be included in the simulator, the layout of a city has to be represented by a specific file containing a set of coordinates: <latitude, longitude, altitude>. These coordinates are extracted by an online crowdsourced tool, such as Digipoint. In fig. 2.3, we can see how this tool works, the use of Digipoint is strictly linked to the precision of the user, who has to manually click on every point for which he wants the coordinates. Therefore, to add new city in CrowdSenSim requires a strong effort.

List of Events Module

This is the core module of the mobility in the simulator. Indeed, it is the module where it is created the list of events. The whole mobility of the simulator is based on the concept of 'Event', it consists in tracking the moves of a user from one position to another and associate them with the corresponding instant of time in which they takes place. An example of an Event is:

$$UID \quad | \quad LAT \quad | \quad LON \quad | \quad DAY \quad | \quad HOUR \quad | \quad MINUTE$$

Where *UID* is the user id, *LAT* and *LON* are the geographical coordinates at the time defined by *HOUR* and *MINUTE*. The simulator uses a sequence of events to reconstruct the path of a user along the city. Afterwards, during the simulation phase, each event is associated to a quantity of generated data. The position of the first event of a user, it is defined by choosing a random location between the coordinates of the city layout. To define the next movements of the user, for each point p_i on the layout, it is created a list of adjacent points, that are the points under a certain distance from p_i, the distance is measured with the haversin formula, the positions reached by subsequent movements are randomly chosen among adjacent

points. For each user it is defined an initial starting time, determined by the User Mobility module, a speed uniformly distributed between [1,1,5]m/s and a total amount of travel time t_{travel}. For every new event is calculated the time t_{next} that the user employs to walk from the previous position to the current, using the distance between the position (calculated with the Haversine formula) and the speed of the user. Once computed t_{next}, it is subtracted from t_{travel}, the user stops moving when $t_{travel} \leq 0$.

User Mobility

In this module is defined the distribution of the users along the day hours, Assigning to each user a corresponding leaving time, in which they start to walk. Two mobility algorithm are implemented in CrowdSenSim :

Uniform Random Distribution It is the simplest mobility algorithm, the initial walking time of users is randomly assigned, it is uniformly distributed between 8:30 AM and 1:30 PM, these are the default hours boundaries but it is possible to modify them.

Time events User Distribution This algorithm was implemented to recreate a feasible human mobility pattern. Each user has a probability to start walking in a certain timeslot that follows a specific probability density function (PDF). An example can be the PDF in fig. 2.4, where the firsts and the final hours of the simulation have an higher probability, and so according to that the users will be distributed mostly along the bounds hours respect to the middle ones.

CrowdSensing Inputs

In order to create the list of Events and during the simulation phase some input parameters are needed, such inputs are provided by a setup file, in which are defined:

- Days of simulation: total number of simulation days. It is a possible value between 1 and 7
- Number of users: amount of pedestrians during simulation

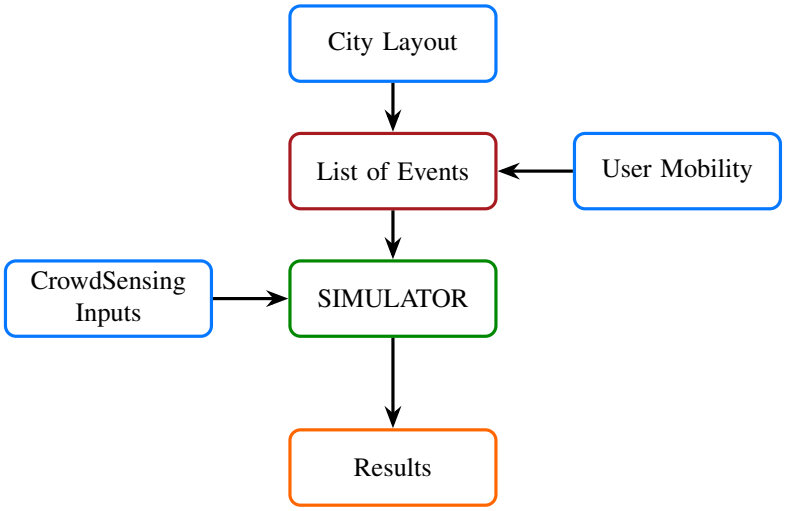


Figure 2.2. Architecture of the old release of CrowdSenSim

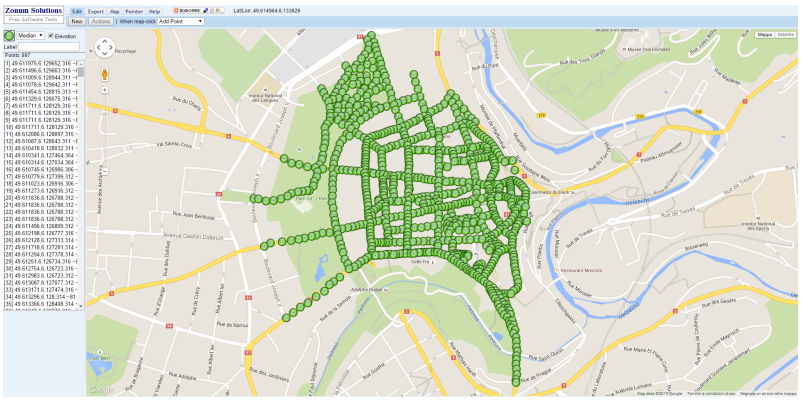


Figure 2.3. Map of Luxembourg obtained with DigiPoint

- Minimum travel time, Maximum travel time: travel time per user uniformly distributed between these values
- Start hour simulation, Start minute simulation
- Finish hour simulation, Finish minute simulation
- Kind of antennas: base antennas system used, allocated in the city
- Create a new list of events: it is possible choose to create a new list of events or using the default one
- Ray: value in meters for the ray useful for list of adjacent

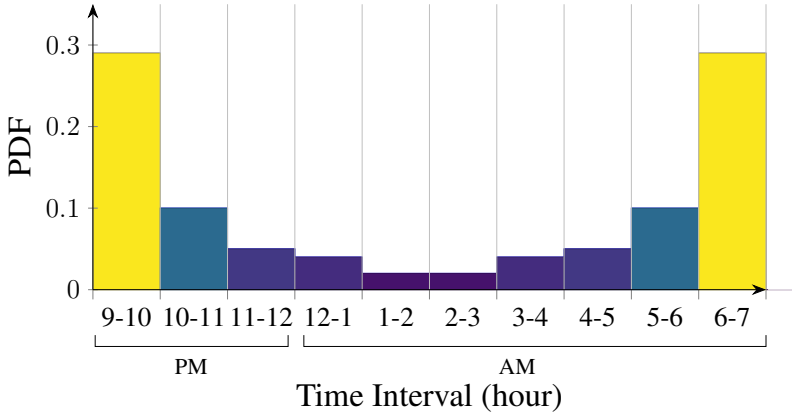


Figure 2.4. One of the possibile user probaility distibution fuction

2.4.2 Weak points and possible improvements

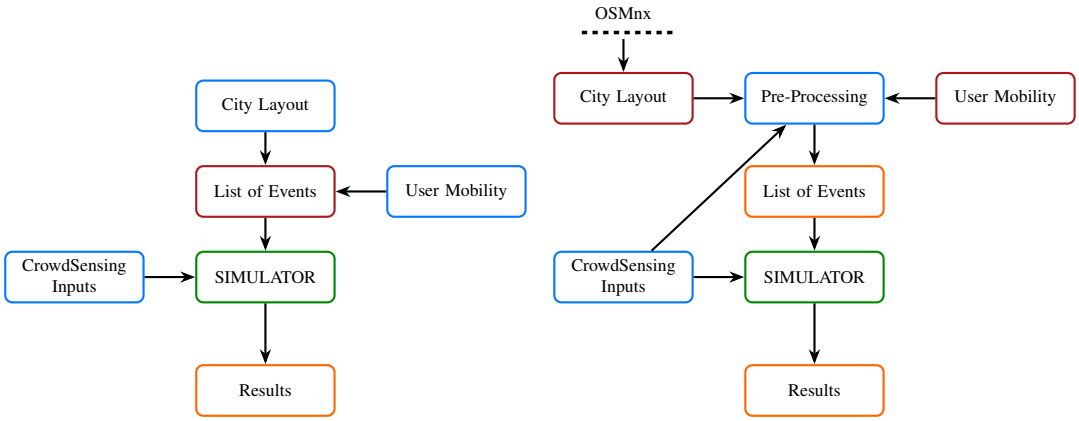
In the old version of CrowdSenSim the weakest module is the mobility. Indeed, a significant problem was that of direction of movement of users, the jumps on the map of each participant were defined only through the list of adjacent, there was the possibility that a user walked only between two adjacent points, doing always the same round trip and repeating it along all the travel period. Another weak point is the city layout, if we run simulations on a city different from the already mapped (Luxembourg, Trento, Madrid), we should manually take the coordinates points of the desired city, using a tool to extract coordinates such as DigiPoint, even though the granularity and precision of the points will be related to the human factor of the person using the tool. A possible improvement concerns the simulation module, in CS0 the participants are sensing along all their travel period and they report the collected data in every timeslot, an a possible alternative is that to develop new data collection framework. An other development for this module would be update the current consumption of devices in each timeslot, in CS0 this value is due to theoretical studies, while an alternative would be include real battery consumption traces.

Chapter 3

CrowdSenSim

I developed a new version of the simulator, in order to solve the issues related to the previously specified weak points,

The structure of CrowdSenSim continues to be strictly linked with the concept of modular architecture. As shown in fig. 3.1 the core layout is the same as before, the biggest difference is the introduction of a new module, the pre-processing. This new module elaborates the new mobility features introduced in this release of CrowdSenSim, during this phase the list of event is created taking account of the inputs parameters obtained from the setup file. Furthermore, an other essential contribution to the simulator is given from the new connection of the city layout module to an extern python package called OSMnx, I will talk about it in the following section. In the first section I will explain the new mobility and its implementation. Afterwards, I will introduce new metrics to analyze simulations. At the end I will describe all the the others features added in the new release of CrowdSenSim.



(a) Old (b) New

Figure 3.1. Architecture of CrowdSenSim

3.1 Mobility Background

3.1.1 OSMnx

In order to obtain and re-utilize maps of a specified place, a researcher had to click through many website and one at the time download all the shapefiles of the required region. To simplify this issue Geoff Boeing introduced OSMnx [6] a new tool to collect and analyze data of street networks. It exploits the graph theory for urban design purpose. OSMnx is a free Python package able to download and analyze geographical boundary, building footprints, and street networks of any location in the world from OpenStreetMap, which is a map created by people under an open license. It allows the user to build and visualize complex graphs, representing street networks of different types: driving, biking, or walking network. In order to be simple and easy to use OSMnx is written in python, many functions need only few lines of code to be executed. OSMnx uses many famous Python's libraries such as NetworkX and geopandas. The main contributes of this tool are:

- the automation of downloading political boundaries and building footprint;
- the construction of graphs based on street networks derived from OpenStreetMap;

- the simplification of network topology, the difference between OSM topology and the simplified version produced by OSMnx is shown in Fig.3.3a and Fig.3.3b
- the possible analysis of street networks,such as calculation of routes and projection of networks.

3.1.2 Mobility in Simulators

Urban planners commonly rely on spatial distribution of citizens or locations to inspect the complex dynamics of urban environments. For example, through analysis of Foursquare checkpoints, Daggit et al. [21] explore how cities grow and highlight spatial correlation as one of the most important growth factors. To determine the most prominent location to open a new activity, Jenses [39] proposed a mathematical model that characterizes the potentiality of locations. For example, the candidate locations opening a bakery are those that mimics the features of the average locations where actual bakeries are located. Another example is [3], where the authors analyze how the growth and positions of craft breweries is related with social and demographic evolution, captured through analysis of the neighborhood change. For urban planning, the sole information of the location of specific points of interest over time such as Foursquare check-ins can be sufficient. Similarly, vehicular traffic simulators like SUMO [4] base the mobility models on spatial vectors and rely on tools like NETCONVERT, which turns a OSM map into a set of nodes defining road intersections and edges defining the roads. In order to be scalable and to obtain meaningful statistical results [23], in MCS simulators the complete path of all users participating and non-participating in the campaign should be known in advance. For example, relying on spatial vectors would increase the amount of computation performed over runtime to determine user positions with fine time precision, lowering the scalability. This is a key element to correctly compute the precise amount of collected data and its associated energy cost under specific policies for data collection [12]. CrowdSenSim defines the trajectory x of each user as a sequence of n steps $x_1, \ldots, x_n$ with $n > 1$, where each of the x_i steps uniquely defines the location in terms of time and spatial references,

i.e., latitude and longitude. This definition is in the spirit of previous research on human mobility [15]. The vertices of street network graph have to be close one with each other to ensure high precision of the sequence of n steps defining user trajectories. The combination of CrowdSenSim with OSMnx would result in a win-win strategy to enhance the flexibility of the simulator. To illustrate, OSMnx-based walking maps would distinguish the two sidewalks of a street, while the previous version of the simulator only sees the street.

3.2 New Mobility

OSM street nodes are inconsistent for direct use in CrowdSenSim because they include dead-ends, intersections and all the points in a segment when streets curve. OSMnx automatically simplifies and corrects topology through an algorithm, removing those points and unifying each resulting set of sub-edges into single edges. The approach is commonly used in different research domains to limit the amount of data to be analyzed [47, 63, 72]. However, for MCS purposes the resulting topology lacks of a sufficiently fine-grained level of details. First, the distance between two points in the graph can be excessively large, making user trajectories to be captured with a minimum of two points. Second, any statistical information on the amount of sensed data would aggregate in a limited number of locations, making the results not very informative To solve the lack of granularity of both OSM and OSMnx nodes i proposed to use a procedure to Augment the OSM Precision (AOP): it builds upon the OSMnx and has been included inside the pre-processing module. i developed an exact solution (V-AOP) based on Vincenty's formula [65] and an approximated one, based on linearized version (L-AOP).

3.2.1 Pre-processing module

As mentioned before, the Pre-processing module contains the main innovations in the mobility of the new CrowdSenSim. It is written in Python and it is built on the OSMnx package. At First it downloads the street network of the desired place,from which consequentially it creates an undirected graph. Afterwards, the

desired granularity and the graph are send as input to the AOP algorithm. The value of granularity has to be between 1 and 10 meters, indeed for larger numbers the algorithm will not ensure the right density of points, this is due to the presence of preexisting nodes downloaded from OSM. Lower granularity will give lower percentage of errors. After the elaboration produced by AOP, the graph is ready for the creation of the list of events. For each user is selected a walking time, uniform randomly distributed between 20 and 40 minutes. In the same way also a speed is associated to the user, this time the random distribution is between 1 and 1.5 m/s. The speed and the walking time are both used to compute the walking distance covered by the user. Consequently, the first step to generate the list of events of a specific user is defining its route. To accomplish this task a random point in the graph is selected as origin of the path. The tough part of this implementation was to find the destination node, knowing the origin and the walking distance. At first I have opted for an approach based on grid division and linear distance. I divided the city layout in grids with an area of 1 Km^2. After the choose of the origin I look at the grid of it, then I pick a grid in the map with a linear distance bigger than the walking distance of the user, this will be choose as grid destination in which i choose the destination point. Once obtained both destination and origin points, I use an OSMnx function called *shortest_path* to find the route on the street network between the two nodes. The problem of this approach is that there is not always correspondence among the linear distance and the network distance. Moreover, there would be the possibility of places too small to provide the required linear distance. To solve this issue I decided to exploit a Networkx function called *single_source_dijkstra*, which discover all networks path in a graph, starting from an origin node and cutting all routes after a certain network distance. The path with the network distance equal to the required walking distance is selected as user route, in case of no paths with the required network distance, it is selected the path with longest length and the process is repeated, taking as origin the destination point of the selected path and subtracting from the walking distance the length of the path.

Once determined the route of the user, a starting walking time will be associated to the user, the methodology of the hour distribution will be explained in the

Subsection 3.2.3. Consequentially, it will be printed out in a text file the list of events of the specific user, for each minute of the walking time will correspond a position of the user route. After all users have printed out their routes the list of events is completed and it is passed to the Simulator module.

3.2.2 AOP

OSM provides the graph of the street network $G_{\text{OSM}} = (V, E)$, where V is the set of vertices or nodes and E the set of edges.

- Each node consists of different attributes, an unique code *OSMID* , the latitude (y) and longitude (x) and a secondary parameter called *highway* which characterizes the road of the node.
- Each edge consists of the following attributes: access, bridge, highway, lanes, maxspeed, name, oneway, osmid, service, tunnel, u, v, width, where u and v are the osmid of the adjacent nodes of an edge.

Let $d(u, v)$ be defined the spatial distance between two generic nodes, being $u, v \in V$. The objective of AOP is to increase the precision of G_{OSM} to achieve a given *target distance D* between any two adjacent nodes. The procedure guarantees that D is also the maximum distance between two adjacent nodes, thus setting smaller D augments the number of nodes in the graph and consequently its granularity. D can be as low as 1 m. By taking as input G_{OSM} and D, AOP generates a new graph G_{AOP}. Note that $|V_{\text{OSM}}| \leq |V_{\text{AOP}}|$. Higher precision means higher granularity, hence the cardinality of $|V_{\text{AOP}}|$ increases.

I developed two algorithms to interpolate the position of the added nodes. V-AOP is based on the exact distance between two nodes in the OSM, and L-AOP is based on a linear approximation of the distance, reducing the computation time.

Fig. 3.3 compares the street network graph of different cities, Brancolino and Luxembourg city. Specifically, Brancolino is a small town in Italy, while Luxembourg is a medium-size city and capital of the homonym country. The figure highlights the different precisions of the street network graph obtained with OSM (see Fig. 3.3(a)), OSMnx (see Fig. 3.3(b)) and AOP (see Fig. 3.3(c)). AOP is

general enough to be applied to any situation where the original precision of OSM is not sufficient.

In the following I will describe the two implementations for the AOP procedure, the first one, V-AOP, exact but computationally complex, and the second one, L-AOP, approximated but less complex.

V-AOP Implementation

The V-AOP version is based on the Vincenty's formula, developed by Thaddeus Vincenty (1975) [65] which assumes the Earth as an oblate spheroid. The Haversine formula [57] approximates the Earth as a perfect sphere, is commonly preferred in GIS applications because is computationally less expensive than Vincenty's exact solution. However, I employ the Vincenty's formula as the AOP procedure operates offline while GIS applications typically run online over computationally-limited mobile devices. Given a node $z \in V_{\text{OSM}}$ over $e \in E_{\text{OSM}}$, it is possible to determine z' over e through the azimuth across u and v so that their distance $d(z, z') = D$. The azimuth between u and v is the angle between a vector that points from u to v and another vector pointing from u to the north (see Fig. 4.5).

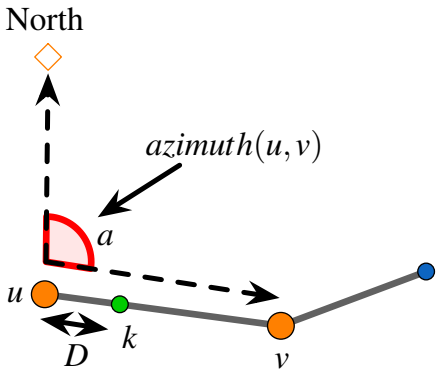


Figure 3.2. Geometry of azimuth determination process

To determine this angle measurement I exploit the trigonometric function *arctan*2, which is a variation of *arctan*. It has two input parameters and is defined

as follows:

$$\text{atan2}(x,y) = \begin{cases} \arctan(\frac{y}{x}) & x > 0 \\ \arctan(\frac{y}{x}) + \pi & x < 0 \wedge y \geq 0 \\ \arctan(\frac{y}{x}) - \pi & x < 0 \wedge y < 0 \\ +\frac{\pi}{2} & x = 0 \wedge y > 0 \\ -\frac{\pi}{2} & x = 0 \wedge y < 0 \\ not\ \ defined & x = 0 \wedge y = 0 \end{cases} \tag{3.1}$$

V-AOP (see Algo. **??**) is implemented as follows:

- For each edge $e \in E_{\text{OSM}}$, V-AOP stores $long_u$, $long_v$, lat_u, lat_u, which are respectively the longitude and latitude coordinates of u and v.

- V-AOP computes the azimuth a from u to v (see Fig. 4.5) as follows:

$$a = \text{atan2}(\sin(\Delta\lambda)\cdot\cos(\phi_2), \cos(\phi_1)\cdot\sin(\phi_2) - \sin(\phi_1)\cdot\cos(\phi_2)\cdot\cos(\Delta\lambda)) \tag{3.2}$$

  where atan2 is a standard trigonometric function commonly implemented in nowadays programming languages, $\Delta\lambda$ is the difference of longitude in radians and ϕ_1, ϕ_2 are respectively the latitudes of u and v expressed in radians.

- Then V-AOP determines the number of points P to create between u and v;

- V-AOP iterates over the edge e by adding an additional node on the edge at each iteration. The position of a new node k is determined with the Vincenty's formula, which takes as input $long_{u,v}$, $lat_{u,v}$, a and D, where D is the target distance between k and u.

  The new node k and the corresponding edge e(u,k) are added to G_{AOP}, while the value of k is assigned to u. After P iterations, the edge is fully characterized and the last sub-edge e(k,v) is added to G_{AOP}: V-AOP moves on the next edge.

Algorithm 1 V-AOP

procedure ADDPOINTS(E_{OSM}, V_{OSM}, D)
  Input: $G_{OSM} = (E_{OSM}, V_{OSM})$, D
  Output: $G_{AOP} = (E_{AOP}, V_{AOP})$
  $E_{AOP} = V_{AOP} = \emptyset$ ▷ Initialization
  for $e \in E_{OSM}$ **do**
    $u, v \leftarrow$ adjacent nodes of e
    $V_{AOP} \leftarrow V_{AOP} \cup$ node($lat_u, long_u$)
    $V_{AOP} \leftarrow V_{AOP} \cup$ node($lat_v, long_v$)
    $a \leftarrow$ azimuth (u, v)
    $L = d(u, v)$ ▷ Distance from u to v
    $P =$ floor(L/D) ▷ Number of nodes to add
    for $i = 1 \rightarrow P$ **do** ▷ For P iterations
      $k =$Vincenty(a,D,$lat_u, long_u$) ▷ New node
      $V_{AOP} \leftarrow V_{AOP} \cup$ node($lat_k, long_k$)
      $E_{AOP} \leftarrow E_{AOP} \cup$ edge(u, k)
      $u = k$ ▷ Assign the new k node to u
    end for
    $E_{AOP} \leftarrow E_{AOP} \cup$ edge(u, v)
  end for
end procedure

L-AOP Implementation

Starting from the observation that for short distances the Earth can be assumed as flat, I proposed a simplified version of V-AOP called L-AOP. L-AOP exploits the Euclidean distance, which only approximates the Vincenty's optimal result, but is simpler to implement and less computationally expensive.

The intuition is as follows: every edge $e \in E_{OSM}$ that is a real street, can be approximated as a straight line l passing through u and v. Considering longitude values on x-axis and latitude values on y-axis, L-AOP determines the equation of l and then through iterations determines points over l with a granularity level D.

L-AOP (see Algo **??**) operates as follow:

- For each edge $e \in E_{OSM}$, L-AOP stores u and v and obtains for both the corresponding latitude and longitude coordinates x_u, x_v and y_u, y_v.
- L-AOP computes the equation of the line l, given by the canonical expression

$y = m\,x + q$, where m is the slope and q is the y-intercept.

$$\begin{aligned} y &= mx + q \\ m &= \frac{y_v - y_u}{x_v - x_u} x_v \neq x_u \\ q &= -(m * x_u) + y_u \end{aligned} \tag{3.3}$$

- Once determined l, L-AOP enters in a cycle with three cases:

  1. $X_u = X_v$;
  2. $X_u < X_v$;
  3. $X_u > X_v$.

- In each of these cases, the objective is to find the coordinates x_k,y_k of a point k along l at a distance D from the original node u of e (see Section 3.2.2 for the definition of D). For this, L-AOP exploits the reverse Euclidean Distance:

$$R = |x_v - x_u| = \frac{D}{\sqrt{1 + m^2}}. \tag{3.4}$$

- In case 1, L-AOP just adds T to y_v and takes it as y_k, while $x_k = x_u$.

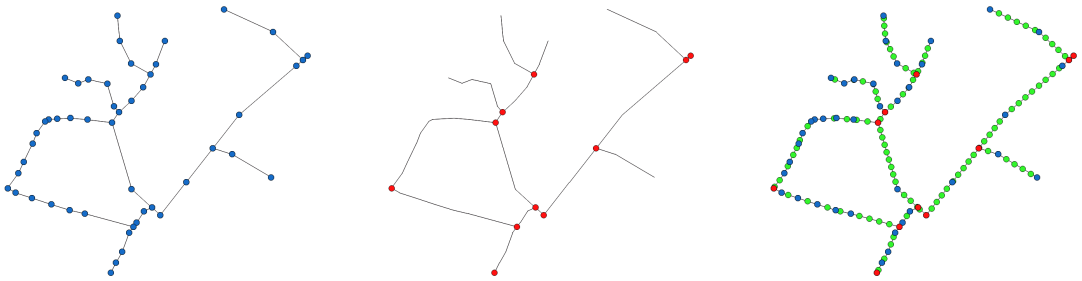
- For cases 2 and 3, L-AOP computes x_k, exploiting the reverse Euclidean distance as follows:

$$\begin{aligned} &\text{case 2:} \quad x_k = x_u + R, \\ &\text{case 3:} \quad x_k = x_u - R. \end{aligned} \tag{3.5}$$

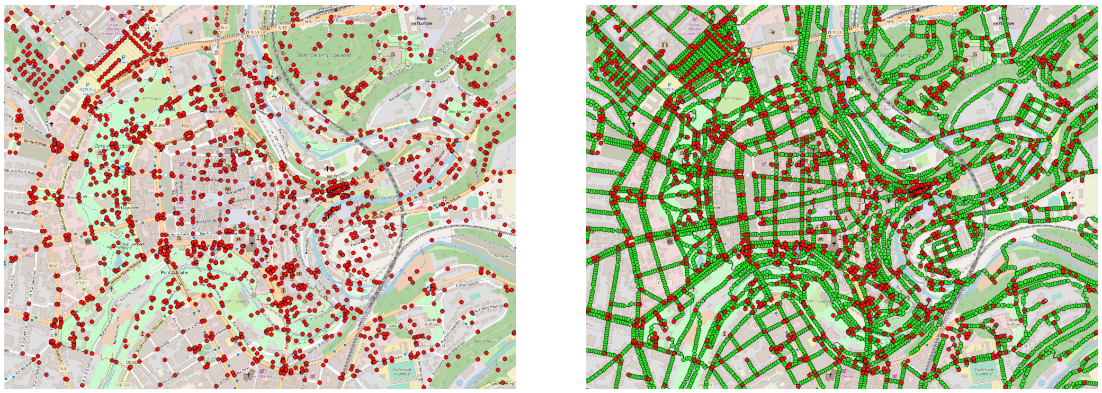
- Afterwards, L-AOP checks if the new coordinate overcomes v. In positive case, it breaks out from the cycle and continues with the next edge. Otherwise, it computes y_k by inserting x_k in the l equation, then the new coordinates are added to V_{AOP} as new node. Finally, L-AOP continues iterating over the same edge for the next node on l.

Algorithm 2 L-AOP

procedure addPoints(E_{OSM}, V_{OSM}, D)
  Input: $G_{OSM} = (E_{OSM}, V_{OSM}), D$
  Output: $G_{AOP} = (E_{AOP}, V_{AOP})$
  $E_{AOP} = V_{AOP} = \emptyset$ ▷ Initialization
  for $e \in E_{OSM}$ **do** ▷ For each edge in original graph
    $\gamma = 110250$ m ▷ Distance in one degree latitude
    $u, v \leftarrow$ adjacent nodes of e
    $y_u, y_v \leftarrow lat_u, lat_v$
    $x_u, x_v \leftarrow long_u, long_v$
    $\beta = (y_v - y_u)/(x_v - x_u)$ ▷ Assume $x_u < x_v$
    $\delta = D/\gamma/\sqrt{1 + \beta \cos(lat_u)^2}$ ▷ Step in x-direction
    $x' = y' = 0$ ▷ Temporary coordinates
    for $x = x_u; x < x_v; x+ = \delta$ **do**
      $y = y_u + \beta(x - x_u)$ ▷ Linear interpolation
      $V_{AOP} \leftarrow V_{AOP} \cup$ node(x, y)
      if $x' \neq 0$ **then**
        $E_{AOP} \leftarrow E_{AOP} \cup$ edge(node(x', y'),node(x, y))
      end if
      $x' = x, y' = y$ ▷ Store the last added node
    end for
    $V_{AOP} \leftarrow V_{AOP} \cup$ node(x_v, y_v) ▷ Add v
    if $x' \neq 0$ **then** ▷ Add last edge
      $E_{AOP} \leftarrow E_{AOP} \cup$ edge(node$(x', y'), v)$
    else ▷ u and v already close less than D
      $E_{AOP} \leftarrow E_{AOP} \cup$ edge(u, v)
    end if
  end for
end procedure



(a) Brancolino - Precision of OSM (b) Brancolino - Precision of OSMnx (c) Brancolino - Precision of AOP



(d) Luxembourg - Precision of OSMnx (e) Luxembourg - Precision of AOP

Figure 3.3. Granularity for different cities. Blue circles denote the vertices provided by OSM, red ones by OSMnx and green circles those created by AOP.

3.2.3 Pedestrian mobility models

CrowdSenSim exploits pedestrian mobility. I developed for the new release of the simulator two user arrival models, which means how the users are distributed along the hours of simulations. In the first model, the user arrivals are uniformly distributed over the simulation period (U-MOB). The second mobility model derives user arrivals from a dataset (D-MOB). Specifically, it exploits real traces obtained from the well-known MCS ParticipAct framework [18]. The duration of each user trajectory and the user walking speed are uniformly distributed between $[20, 40]$ minutes and $[1, 1.5]$ m/s [23] respectively. The active users move in a random walk over the street graph G generated by the AOP procedure. This section provides details about the two models.

The U-MOB Mobility Model

With U-MOB, user arrivals are uniformly distributed over the simulation period, i.e., the average total number of users that start walking over a certain time window is constant. Typically the time window is 1 hour long for comparison with D-MOB (Subsec 3.2.3), as ParticipAct traces are given with such granularity. Users who start and end their walking period in different hours are counted only once. At the start of pre-processing module it is computed the expected number of users per hour. During the allocation of users when the expected number is reached the next user will start walking in the next hour.

The D-MOB Mobility Model

Chessa and al. [18] compare different human mobility datasets. In this work, I focused on the ParticipAct dataset, originated from a MCS campaign of around 170 students in the Emilia Romagna region (Italy). Without having at disposal the dataset, I extracted the profile of the average number of contacts during 7 days and used as a reference to determine the user arrivals for D-MOB.
Specifically, given the total simulation period in days, I subdivide the period into hours and then estimate the minimum number of users to be allocated so that the average user contact follows the ParticipAct profile.

Note that a user contact occurs when two users are within a certain distance R. For simplicity, I decided to count unique contacts even when users trajectories intersect multiple times.
For D-MOB, the following two methodologies have been designed:

- The Contact-Only-Distribution (COD) assigns a number of users per hour to reach the corresponding average number of contacts from the dataset. Once this preliminary phase is completed, the remaining not assigned users are divided proportionally to the per hour number of contacts. The method favors hours with high number of contacts.

- The Contact and User-Distribution (CUD) method is hybrid. It operates similarly to COD for the first phase, but in the second phase it assigns the

remaining number of users proportionally to the number of hours of the simulation period. This method favors equally hours with high and low number of contacts.

Both methodologies operate the preliminary phase. Specifically, it consists in the allocation of users in a fictitious hour where the expected number of users is due to the average percentage of contacts of all hours. Once the factious hour is fulfilled with the expected number of users, it is saved the number of contacts occurred. The number of contacts expected during the allocation of users in the real hours will refer to their corresponding percentage of contacts with reference to contacts of the fictitious hour.

Accuracy of the D-MOB arrivals methods. To evaluate the accuracy of the methods employed to generate the user arrivals according to the ParticipAct dataset, I ran two different experiments using Luxembourg as city of interest.

In the first experiment, I put four scenarios with increasing number of users in the system, and the simulation period is fixed to 12 hours. Fig. 3.6(a) shows the relative error while comparing the methods COD versus CUD for D-MOB. Note that the accuracy is high as the error remains lower than 0.25% and is nearly constant for different values of users in the system. Note that the spike achieved for $20,000$ users is negligible and due to the different initial spatial allocation.

In the second experiment, I evaluate the accuracy by increasing both the number of users and the simulation period. Fig. 3.6(b) shows the results for different increase factors. An increase factor equal to $1\times$ corresponds to $10,000$ users in the system and a simulation period of 24 hours. An increase factor equal to $4\times$ corresponds to have $4\times$ the baseline number of users and simulation period. Hence, $4\times$ corresponds to $160,000$ users over 96 hours. The results are pretty accurate and the highest error is nearly $0.4\,\%$. Unsurprisingly, the highest error occurs for the lowest number of users in the system as achieving the target number of contacts is more difficult and can only be alleviated by increasing the population in the system.

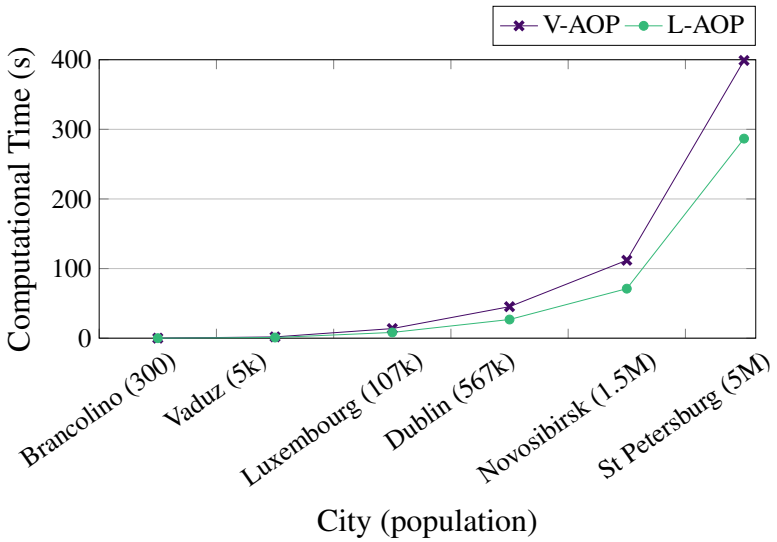
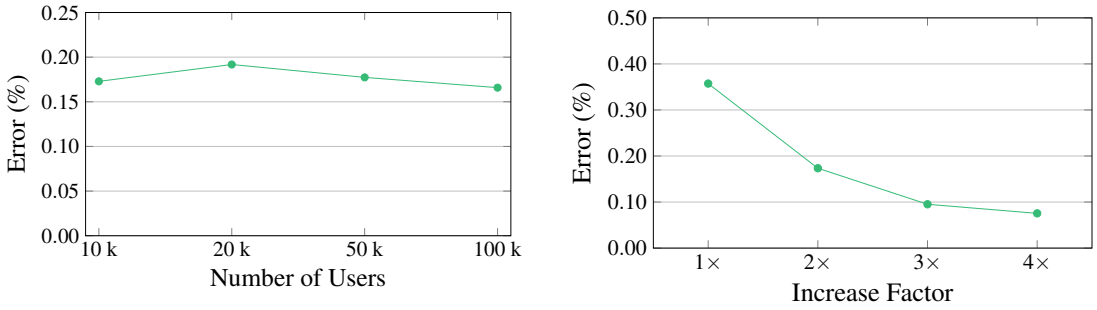


Figure 3.5. Computational time and relative accuracy of V-AOP and L-AOP



(a) Simulation period fixed at 48 hours

(b) Joint increase of users and simulation period

Figure 3.6. Accuracy of mobility models

3.3 Novel key performance indicators

In this section i detail the new metrics that I inserted in CrowdSenSim. They differ for how they evaluate different aspects of the simulation. First part introduces metrics to analyze the new mobility, while in the second part are provided metrics about fairness between users.

3.3.1 Human mobility metrics

In order to assess U-MOB and D-MOB arrival models, I decided to implement two new metrics for CrowdSenSim . For both metrics I simulated a population of 50,000 users distributed over a period of 48 hours. I assume the time is slotted, with a timeslot equal to 1 minute. Two users are defined as a neighbors if during timeslot j their distance is below a given radius R. For the experiments, R is set equal to 50 m, which is a reasonable distance for D2D communications based on WiFi-Direct. The per-user average number of contacts UAC, specifically evaluated for user i, is defined as follows:

$$\mathrm{UAC}_i = \frac{1}{T_i} \cdot \sum_{j=1}^{T_i} n_{j,i}, \tag{3.6}$$

where T_i is the amount of time user i is active, measured in timeslots, and $n_{j,i}$ is the number of neighbors of user i in timeslot j.

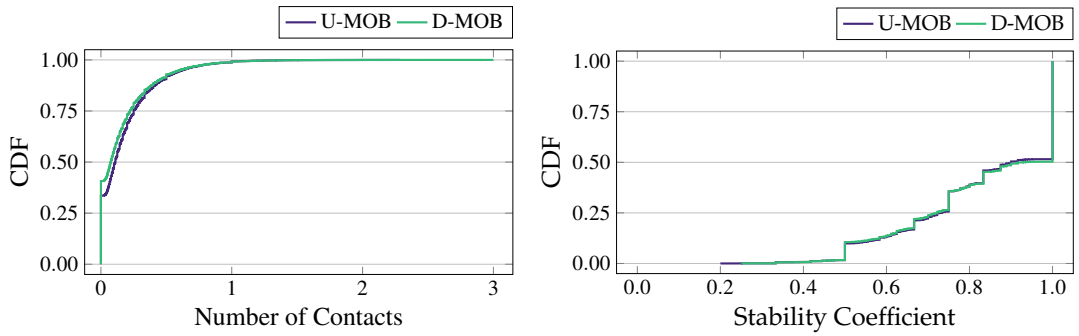
The second metric is called stability coefficient SC [8] and for user i is determined as follows:

$$\mathrm{SC}_i = \frac{1}{T_i} \cdot \sum_{j=1}^{T_i} \frac{|n_{j+1,i} \setminus n_{j,i}| + |n_{j,i} \setminus n_{j+1,i}|}{|n_{j,i}| + |n_{j+1,i}|}, \tag{3.7}$$

where $n_{j,i}$ denotes the set of contacts of user i during timeslot j, $|n_{j+1,i} \setminus n_{j,i}|$ is the number of neighbors that user i loses between timeslots j and $j+1$ and $|n_{j,i} \setminus n_{j+1,i}|$ corresponds to the number of neighbors that users i acquires between timeslots j and $j+1$. This metric determines on how frequent a user changes neighbors, hence is relevant to determine to which degree the user is a valid candidate to become group owner in D2D WiFi-Direct communications.

Fig. 3.7 illustrates the results in form of CDF (Cumulative Density Function) for the distribution of the metrics evaluated across all the users. Fig. 3.7(a) shows the UAC metric. Interestingly, with the considered high number of users in the system, the U-MOB method with uniform arrivals approximates pretty well the D-MOB arrival distribution, based on the ParticipAct data set. However, it should be noted that the distribution of UAC metric depends on multiple factors, including the total number of users in the system, the duration of the simulation, the size of the urban environment and the spatial allocation policy, which is uniformly

distributed in our case. Fig. 3.7(b) indicates that half of the users in the system obtains values of SC different from 1, hence between two subsequent timeslots the users remain in contact with their neighbors. On average, for the majority of the users (75 %), nearly 30 % of the contacts is stable as the SC metric assumes values below 0.7.



(a) Per-user average number of contacts (UAC)

(b) Stability coefficient (SC)

Figure 3.7. Analysis of contact distribution and stability of contacts

3.3.2 Fairness

Ideally, MCS systems should gather information from the crowd by ensuring QoI. At the same time, 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. This is the reason why I opted for the implementation in CrowdSenSim of a new fairness metric.

To evaluate fairness between users, I decided to exploit the Jain Fairness Index [38], which measures the equality of user allocation as follows:

$$F_J(x) = \frac{\left(\sum_{i=1}^{N} x_i\right)^2}{N \cdot \sum_{i=1}^{N} x_i^2}, \tag{3.8}$$

where N is the number of users and x_i is a generic resource allocated to user i. The

quantity x_i changes according to the allocation metrics, which depends on the purpose of the study. If the objective is to distribute workload equally between all users, x_i indicates the allocation to user i. Otherwise, $x_i = \alpha_i/\beta_i$ aims to allocate the amount of total contribution with different weights, where β_i is a weight to share the resource to user i and α_i is the allocated resource. In the first approach, the index is 1 and the system is totally fair when all x_i values are equal. For instance, in [46] x_i are the observations in a region of interest. In the second approach, the fairness index assumes values equal to 1 when the resources are divided according to different weights. For instance, if a user pays the double than other users for a shared resource, the allocation algorithm gives twice as much as others to be fair. I follow the second allocation metrics, exploiting different parameters x_i that correspond to a system allocating unequal fractions and corresponding to different expected contributions from users. I adopt two different approaches, one considering fairness as contribution of data, the other one considering battery drain.

Intuitively, users that walk for longer time periods are expected to contribute more data than others. I define the data contribution fairness index (F_{D}) as follows:

$$F_{\mathrm{D}} = \frac{\left(\sum_{i=1}^{N} d_i\right)^2}{N \cdot \sum_{i=1}^{N} d_i^2}, \tag{3.9}$$

where:

$$d_i = \frac{D_i}{D_i^M}. \tag{3.10}$$

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, I 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:

$$F_{\mathrm{B}} = \frac{\left(\sum_{i=1}^{N} b_i\right)^2}{N \cdot \sum_{i=1}^{N} b_i^2}, \tag{3.11}$$

where:

$$b_i = \frac{B_i}{B_i^T}. \tag{3.12}$$

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.

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

$$F_{\mathrm{CS}} = \sigma \cdot F_D + (1 - \sigma) \cdot F_B, \tag{3.13}$$

where σ 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.

3.4 New implemented features

This section contains all the others features added in the new release of CrowdSenSim. At first I start talking about the new data collection frameworks (DCF), which defines the set of steps necessary to produce and deliver the information from the participants to the collector, consequentially I will introduce the novel data reporting mechanisms (DRM), which are the core of any DCF. Afterwards, Subsection 3.4.3 explains how I inserted real smartphone battery traces to profile real energy consumption. Finally I will show the renewed Input-Output of CrowdSenSim.

3.4.1 Data collection frameworks

The organization of a MCS campaign requires to sustain costs to reward individuals for their involvement and to verify the accomplishment of the tasks. Consequently,

it is crucial to investigate how to maximize the efficiency of a Data Collection Frameworks (DCF), which is defined in terms of the costs sustained by the organizer and the revenues. The users sustain costs while contributing data too, i.e., the energy spent from the batteries for sensing and reporting data and, eventually, the data subscription plan if cellular connectivity is used for reporting.

Developing efficient DFC is crucial to regulate the degree of user involvement and to prevent excessive battery drain from the mobile devices, which are fundamental limiting factors to foster user participation and contribution. At the same time, a DCF has to gather a sufficient amount of data to ensure Quality of Information (QoI) .

I implemented in CrowdSenSim three different DCFs. Each of them has specific properties and features that are presented in the followings.

DDF - Deterministic Distributed Framework.

DDF is a DCF with the objective of fostering energy-efficient data collection in opportunistic cloud-based MCS systems proposed in a former work [13]. 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 sends periodically beacons to the mobile devices to share the utility in receiving data from specific sensors. Then, the mobile devices take sensing decisions locally and independently one with each other. Sensing and reporting decision are driven by several factors such as the environmental context, the potential cost for sensing and reporting, the current level of battery and the amount of data already contributed. When these factors are above predefined thresholds, the systems prevent users from further contribution. Therefore, this mechanism is aware of the *user-state*, i.e., it considers the user history to drive the subsequent decisions. The DCF aims to be fair in terms of the amount of energy spent by the users. To implement this DCF in CrowdSenSim, i had to make a battery update procedure in every minute of the simulation, in such a manner every users in each instant have the corresponding actual battery level. Therefore, using DDF in the campaign, all users will continue sensing and reporting data checking every minute their actual battery level and

comparing it to the starting level, in case of level of battery consumed is greater than the pre-specified threshold, the user will stop generating and reporting and will exit from the active status for the remaining minutes of his walk.

PDA - Probabilistic Distributed Algorithm.

Montori et al. [46] propose a distributed algorithm based on probabilistic design to acquire data in an opportunistic scenario. The algorithm is based on limited feedback from the central collector and does not require users to complete specific task. The objective of this algorithm is to regulate the amount of data contributed from users in a certain region of interest to minimize data redundancy and energy waste. Assuming that it is impossible to compute the number of participants in a region of interest because their position can not be disclosed for privacy reasons, the coordinator determines the number of participants through the number of observations received. The ultimate objective is to control the number of observations in an area in order to guarantee a desired value, i.e., the minimum quantity of data to be harvested. To reach this goal, the mobile devices decide independently from the collector and one with each other when to perform sensing and reporting. The framework is memoryless. Unlike DDF, it does not take into account the previous status of each user to determine the probability to report data in a certain timeslot. Each timeslot has a probability P_t to asses if during it, the device will deliver data to the Data Collector or not. In PDA, the users continuously generate data and every minute they determine the probability of delivering the acquired data. When no transmissions occur, because P_t is lower than the threshold probability P_{th} data is stored locally on a buffer whose occupancy increases and decreases with the number of successful delivery attempts.

PCS - Piggyback Crowdsensing.

PCS [41] is a DCF that reports data only during the so called smartphones opportunities, data is delivered during users phone calls. The main advantage related to PCS is the lower energy consumption associated to reporting operations.The reason of this is that the mobile devices do not have to wake up the radio interface

to transmit the collected data on purpose. PCS does not include any feedback from the collector to the users, hence it can not indicate urgency for additional data. Furthermore, if smartphones opportunities are rare, the collector might suffer because of missing data. According to these considerations, PCS is particularly suitable for delay-tolerant MCS tasks that do not need data to be sent to the central collector as soon as users sense it. PCS implements a buffer mechanism as well to store the acquired data that is delivered during phone calls. The distribution and the duration of daily phone follow the profile of weekday 1 in [68], that is computed by normalizing the average call arrival rate and average calls duration within 24 hours from a dataset of four different days

Collected data distribution

In order to show the different amount of data obtained in MCS campaign based on different DCFs, I ran a simulation on CrowdSenSim for each DCFs and I will introduce the results in the following.
Fig. 3.8 illustrates the trajectories of 5 participants walking in Luxembourg City. The aim is to focus on the periods of active contribution and show the differences between the DCFs. DDF allows the users to perform continuous contribution until the mobile devices of the participants meet the conditions to stop to contribute data. PDA shows an intermittent reporting, which is attributed to the probability of performing data transmission. For example, the sections of the trajectories when users perform continuously active contribution depend on the feedback of the collector that asks for additional data. Hence, for all the users, the probability to transmit is high. With PCS, the user active contribution is extremely limited and totally depends on the probability of performing user calls and their duration.

Fig. 3.9 shows the spatial distribution of the total amount of collected data at the end of the simulation period for Luxembourg city. 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. 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.

3.4.2 Data reporting mechanism

A DRM is the most important component of a DCF that defines the methodology to perform delivery of sensed information to the cloud collector. Note that 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)

Data can be 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 DDF. It is useful for all applications that need real-time data even if it could represent a higher cost to contributing users. It is typically the most expensive approach in terms of battery drain and it is utilized in a pure way only if strictly needed. Some constraints may be applied to make it less consuming (e.g., data collection through context recognition).

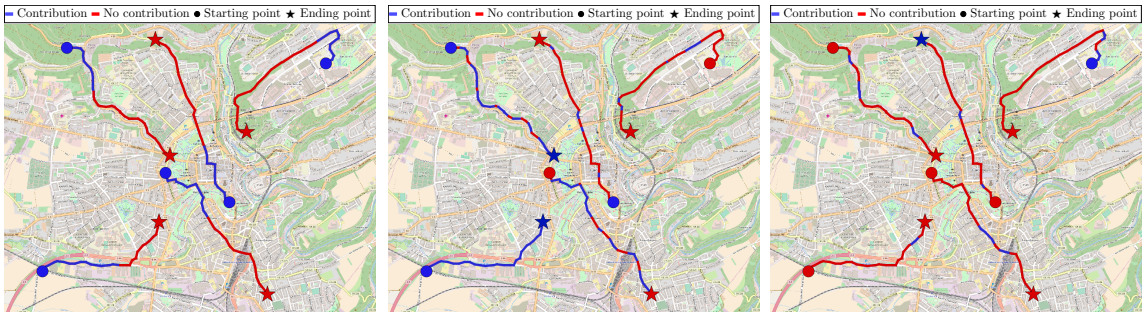
Delayed-DRM (DEL)

With delayed reporting, data is sent 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 interfaces do not have to be maintained continuously active for a prolonged period of time. DEL is useful for delay-tolerant applications where the collector does not

need to harvest data in real-time. DEL is implemented by PCS when the number of smartphone opportunities equals 1.

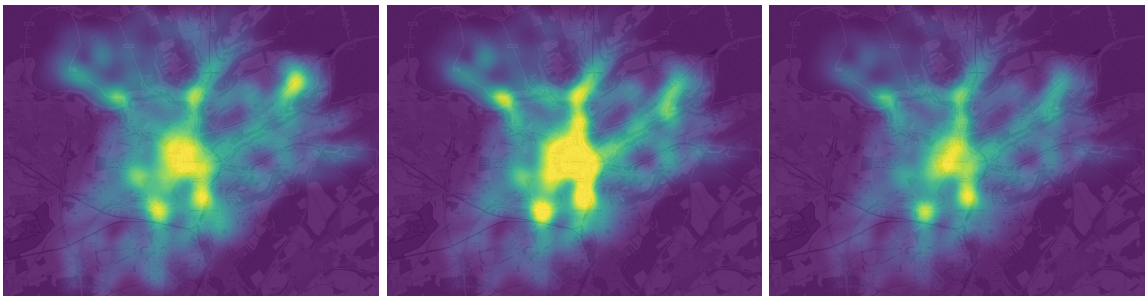
Probabilistic-DRM (PRO)

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 the checked against δ, a feedback provided by the collector. This DRM is implemented by PDA and by PCS when the number of smartphone opportunities is higher than 1.



(a) DDF (b) PDA (c) PCS

Figure 3.8. User trajectories with the associated data contribution in Luxembourg City



(a) DDF (b) PDA (c) PCS

Figure 3.9. Heatmaps of Luxembourg city with different DCFs

3.4.3 Real traces of smartphone energy consumption

In order to improve the assessment of the energy consumption for a sensing campaign in real urban environments, I included in CrowdSenSim real traces of energy measurements for sensing and reporting.

To profile Energy Consumption I used energy measurements obtained at the University of Luxembourg using a crowdsensing Android application.

The app collect data of the same smartphone sensors considered by CrowdSenSim (GPS, Proximity, Accelerometer), the reporting infrastructure is the WiFi and it implements the various DRMs.

The mobile application can run over any Android-based smartphone. Java is the programming language employed for the implementation, while PHP is the server-side scripting language used for the web development. The minimum supported version is Android Marshmallow 6.0 (API level 23), The server side, i.e., the cloud collector is a laptop used to perform data processing and storage.

The energy measurements were performed on the smartphone through a power monitor. Specifically, the cost that each device experiences is computed proportionally to the time of contribution. The reference power consumption profiles were obtained from 30 min long sensing traces. Fig. 3.10 shows the CDF of the energy spent on the various DRMs along all the 30 min experiment.

In order to insert such power monitor results in CrowdSenSim, I computed an average per minute battery consumption, I obtained it exploiting the CDF of the experimental results and making a translation to convert mA measure to a mA h value. Consequentially in CrowdSenSim the battery of each contributing smartphone is updated every minute of traveling, decrementing from the current battery value, expressed in mA h, the value of battery consumption of the corresponding DRM adopted in the simulation campaign.

The decision about use or not the real battery traces for the simulation is easy to change through the input parameters of CrowdSenSim.

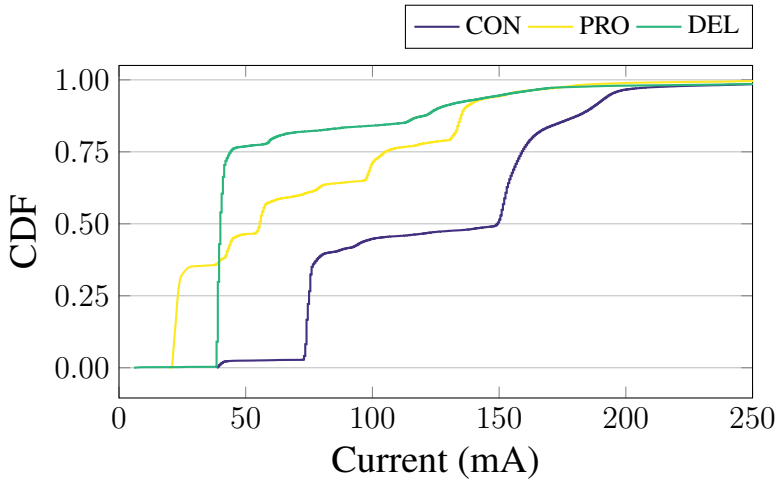


Figure 3.10. CDF of energy spent for different reporting approaches

3.4.4 Input and output improvements

3.4.5 GUI

An important update for the Input of CrowdSenSim is the development of a Graphical User Interface. As before mentioned in Section 2.4.1, in the old versions of CrowdSenSim all the inputs configurations were managed by a setup text file. The Simulator-User had to fill it out with the desired parameters, in order to run the simulation with inputs values different from the default ones. Fig. 3.11 shows the first draft of the CrowdSenSim, it is very simple and easy to use, it is opened to many future improvements. Starting from the top left, the first box is empty for future options, the second shows the actual number of users and days, the third box permits to change such numbers by inserting the desired values and by clicking the ok button, the maximum number of days allowed is 7 while there is no limit for users. The first box on the bottom left is where is possible specify the name of the city or region to simulate, in this box are present two button one is called save and the other save and run. The new release of CrowdenSim is able to save list of events and allows the users to reuse it in next simulations, indeed the two buttons are positioned in case an user wants to save the list that the simulator will produce or just save it for future needs without running the simulation. The second box is a catalogue of the list of events already saved in simulator, in case an

user wants to run one of it, he has to select it and then click the run button, there is also the possibility to delete one list from the catalogue pressing the button del, in addition is possible to check the number of users and days of the selected list looking at the frame at the bottom of the box. The third and last box contains the run button that starts the simulation.

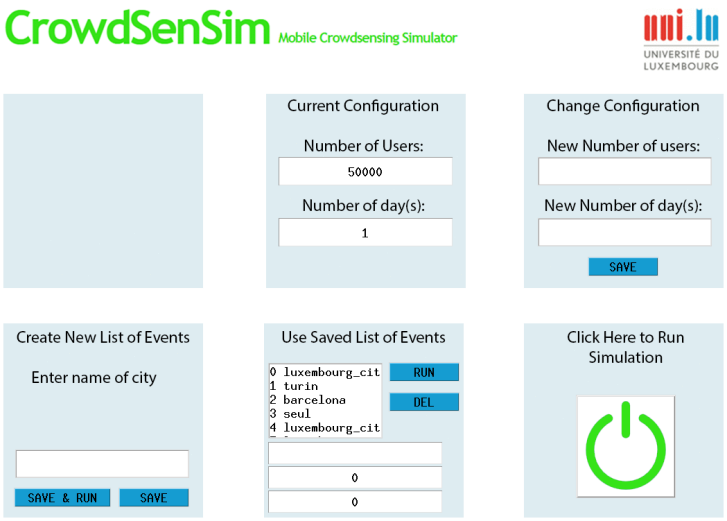


Figure 3.11. Graphical User Interface

Output

The statistics and output maps and graphs are shown at the end of the simulation phase, in a HTML page. Fig.3.12a shows the first part of the page, the table at the left bottom part contains the *Settings of simulation* .

The table in the top right part contains simulation statistics such as the *Average number of samples generated*, related to all the type of sensors. *Average amount of data generated*, represents the amount of data generated during the days of simulation, *Average amount of battery drain spent for sensing*, measured in mA h, counts the amount of battery consumed for sensing during the simulation, by all the pedestrians smartphones. Between the two tables it is positioned a new element for CrowdSenSim Outputs, the route of user, in particular I decided to show the route of the first user during the first day of simulation. The path is drawn above the open street map and two markers are located to define the starting and

ending points The bar chart at the bottom illustrates the *Average amount of battery drain due to the data transmission per-user (*J*)*, ordered by day. In details, the previous parameter represents the energy spent for the communication to the collector of the sensing data collected during the simulation. Other studies underlined how the *Average amount of current spent for sensing* is negligible comparing with the energy spent for data reporting [11] [13].

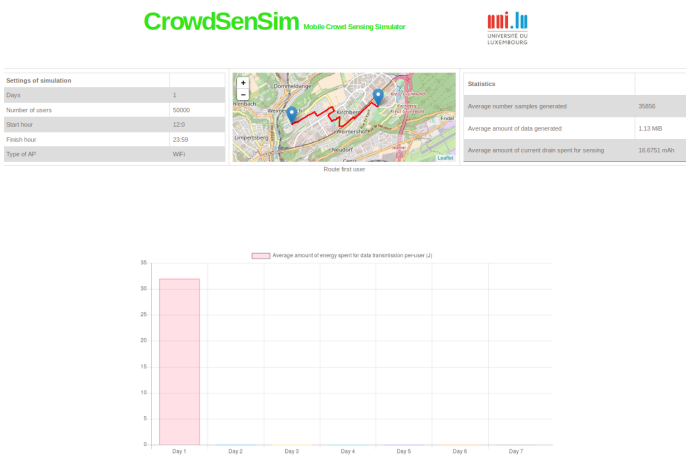
In Fig.3.12b there is the second part of the HTML output page. The first map at the top is the new heatmap of contributing users, it is related with the time of the simulation, this because the heatmap changes every second showing the total distribution of generated data during the first hour of simulation. The area with an highest amount of reported data are denoted in red.

The pie chart in the bottom-left reports the information concerning the *Statistics users sensors*. In details, are shown the quantities of data generated by the three sensors used by default in CrowdSenSim as *GPS*, *Proximity* and *Accelerometer*. The reported quantities are strongly dependents by the sampling frequency of each sensor. The Table 3.2 shows the sensors equipment parameters using for performance evaluation in the default scenario.

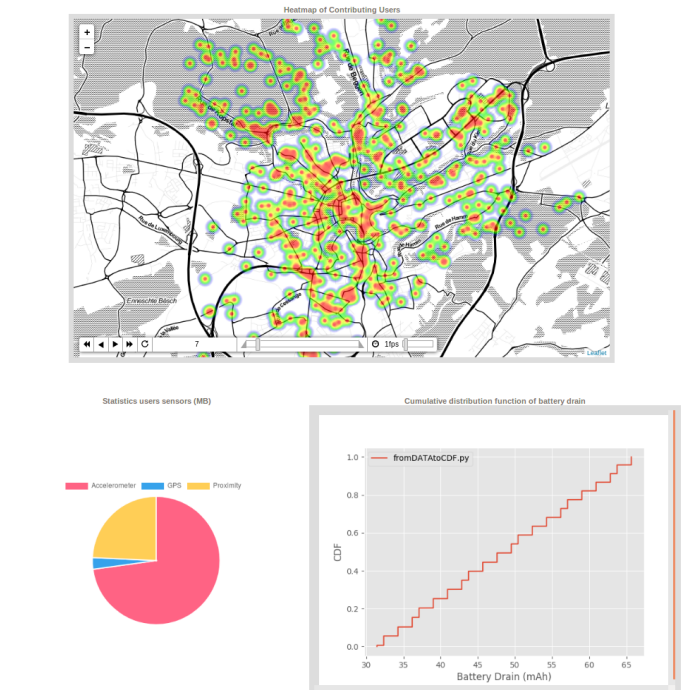
At the end in the bottom-right is included a new figure showing the *Cumulative Distribution Function of battery drain*,which reports the CDF of the total battery drain per user.

Sensor	Parameter	Value	Unit
Accelerometer	Sample rate	4	kHz
	Sample size	6	Bytes
	Current	450	μA
Proximity	Sample rate	8.1	MHz
	Sample size	2	Bytes
	Current	150	μA
GPS	Update period	10	s
	Sample size	24	Bytes
	Current	23	mA

Table 3.2. Pre-configured sensors



(a) First page



(b) Second page

Figure 3.12. Output web-page with results

Chapter 4

Collaborative sensing framework

Typically users contribute data to MCS systems individually and independently from others. On one side, it implies from the central authority the allocation of tasks to each participants. On the other side, they should be able to report data through available communication technologies, such as WiFi or cellular data interfaces. Allocating task in a fair and efficient way is one of the most crucial requirements in a sensing campaign. To accomplish a task, a user has to follow different steps and allocation schemes Chatterjee et al. [16] investigate the main issues in assigning tasks and propose efficient schemes that meet application requirements under. iCrowd [70] is a framework which aims to optimally allocate tasks considering incentive mechanisms under different time and space coverage constraints. Specifically, it maximizes the overall k-depth coverage with a fixed budget or to minimize the overall incentive payment while ensuring a predefined k-depth coverage constraint. ActiveCrowd [29] is a framework to select workers for multi-task campaigns. It aims to minimize the distance a user has to move to the task intentionally.

In the last years, researchers have mainly focused on energy efficient data collection that exploits individual reporting. To give a few examples, Piggyback CrowdSensing [42] is a framework that leverages phone calls and usage of mobile application to deliver sensed data, lowering energy consumption on individual mobile devices. In [13], the authors propose a framework that maximizes the utility of collector in receiving data while minimizing the energy consumption of

smartphones, which deliver information individually. In MCS systems, exploiting collaboration between users to efficiently report data still remains a field that requires further investigations.

Collaborative sensing is based on the idea to form a group between users in proximity and electing an owner, who is the only responsible to deliver data to the central collector. Mobile devices within a group can be seen as nodes or peers in the same network. They require only the knowledge on local neighborhood to exchange data and select the next hop, which depends on location and mobility of users. This mechanism can be seen as a position-based routing over ad hoc network. As forwarding decisions require a precise knowledge of current neighborhood not based only on forwarding node's perception, all suitable neighbors of the forwarding node are involved in the selection of the next hop [26]. Users exchange data through device-to-device (D2D)communications exploiting short range technologies, such as Bluetooth, WiFi direct, LTE direct, Zigbee, near field communication (NFC) or ultrawide band (UWB) technologies. To illustrate, De Benedetto et al. [5] propose to leverage LTE direct for D2D proximity discovery and local data dissemination in next generation 5G networks. WiFi Direct is another technology that provides the possibility to deploy opportunistic networks between users that want to exchange data [19]. Asadi et al. [2] introduce a channel-opportunistic architecture built on top of D2D features of 5G networks that exploits D2D communication. They aim to improve the user experience in terms of throughput, fairness, and energy efficiency, focusing on mobile devices that exploit both WiFi and cellular interfaces to establish a D2D communication. In D2D communication, users need efficient content discovery mechanisms to forward their queries to the node in the neighborhood that is most likely to satisfy them, without a central coordination authority. Casetti et al. [14] investigate this problem exploiting WiFi Direct and proposing intra- and inter-group communication methodologies. In addition, they investigate the group formation, aiming at achieving full connectivity and efficient resource utilization. The policy adopted to form a group is crucial in collaborative MCS systems. In [66], the authors present a protocol that aims to enhance the user performance from an energy and throughput perspective, managing the group size and the transmission power of

each smartphone. Zhang et al. [73] investigate the limitations of WiFi Direct in forming groups with more than two users, presenting a protocol for larger groups. After group formation, it is crucial to elect a responsible of the group that manages it and delivers data to the central collector. The decision to elect an owner is typically left to the application layer. Jahed et al. [36] propose an optimized group owner selection, which aims at enhancing the overall throughput.

In this Chapter, I investigate a collaborative framework that aim to form groups of users who communicate and exchange data through WiFi-direct interfaces and elect a group owner as the responsible for data delivery. At first I introduce the standard of WiFi Direct, afterwards I propose three different grouping strategies and compare them according to the requirements of the MCS campaign under analysis, at the end I present the results of simulation obtained through CrowdSenSim in order to evaluate the proposed framework.

4.1 Background WiFi Direct

Overview

Wi-Fi Direct represents a standard capable to enable Wi-Fi devices as, for example, smartphones, laptops, Smart TVs, printers, cameras and other tools to interconntect in a rapid and easy manner without the integration of an Access Point (AP). Wi-Fi Direct is based on WLAN frame structure and is secured with Wireless Protected Access – 2 (WPA2). It supports, as in normal Wi-Fi, a high data rates up to 250 Mbps and its range connection reaches up to 200m (this contitutes the theorical range; in fact practical range might be different). The main Wi-Fi Direct feature are descriped in the following.

Network architecture

The core concept of Wi-Fi Direct architecture is the “WD Group”. A WD Group is made up of a Group Owner (GO) and zero or more Clients. The GO takes the role of a legacy AP, implementing all its functions within the other group members. A device can dynamically take the role of GO or client. The roles of Devices are

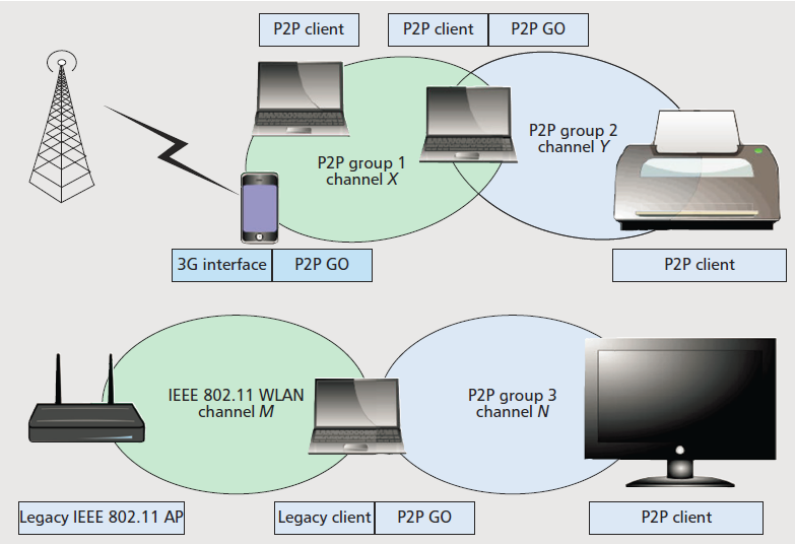


Figure 4.1. Example of Wi-Fi Direct use cases. Reproduced from [9]

usually negotiated before creating a WD Group and remain fixed while the Group is active. No changes of roles are possible along the life period of a WD group.

Device Discovery

In order to form a WD Group, a Device needs to scan for other devices, that are close to it in its wireless range and that are willing to step into a WD group. To do such neighbor research the devices should execute a Device Discovery . The procedure consists of two distinct phases: Scan and Find. In the Scan phase, the Device simply performs a traditional Wi-Fi scan through all supported channels in order to discover existing WD groups, legacy access points and nearby devices with their informations such as the Device Name. Once the device completes, the device steps into a Find phase. In the Find phase, the device chooses one of the predefined social channel 1, 6 and 11 in the 2.4 GHz band, next the device alternates between two states: Search and Listen. In the Search state, the sends Probe Requests on the social channels . In the Listen state, the device waits for Probe Request from other devices, listening on the previously chosen social channel. In case the listening device receives a Probe Request, it will reply with a Probe Response. The amount of time dedicated to each state is randomly chosen. The Device Discovery process can induce some delay to let a Device discovers all

Devices in its vicinity. This delay, can be significant high in case of many Devices simultaneously performing Device Discovery in the same wireless range.

Service Discovery

Service Discovery is an optional feature in Wi-Fi Direct. The procedure starts after the Device Discovery and prior to the Group Formation procedure. It allows a device to query for services offered by nearby devices and to connect only to the ones that makes available a specific service. The Discovery procedure is made possible by the link layer Generic Advertisement Service (GAS) protocol, that allows to transport higher layer protocols. Wi-Fi Alliance has defined a set of standardized services supported by Wi-Fi Direct such as Play, Send and Print.

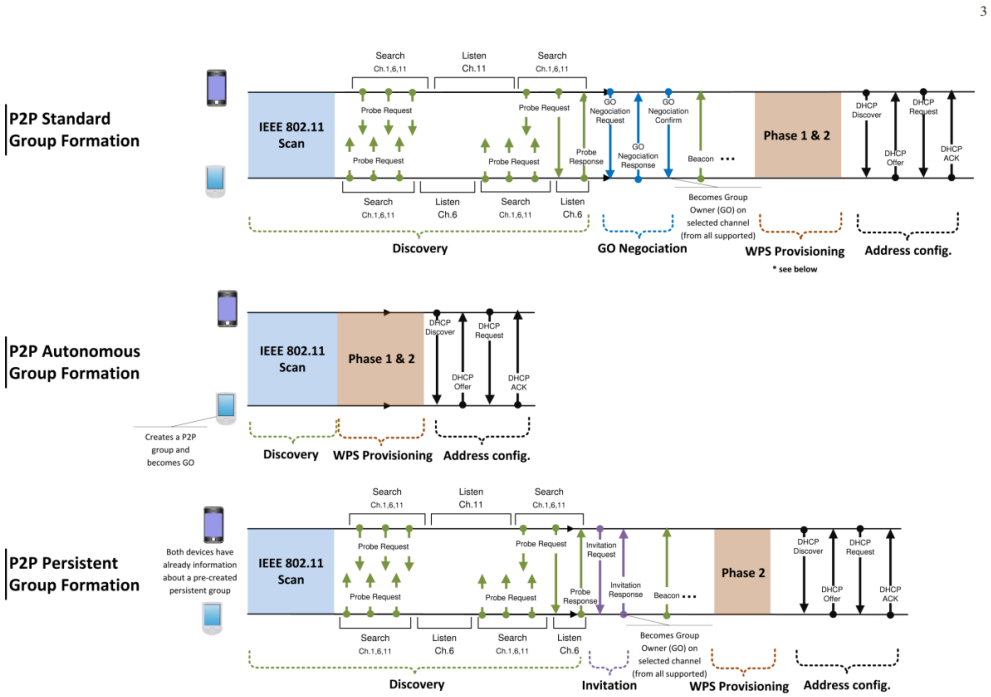


Figure 4.2. Standard Group Formation Mechanism. Reproduced from [9]

Group Formation

In this phase there is the real establishment of the WD Group. During the Group Formation, the device that will act as GO is determined. There are three different

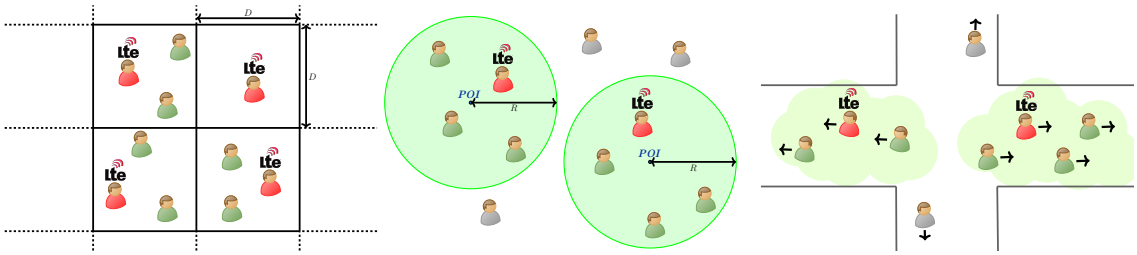
way to form a group: Standard Group Formation, Autonomous Group Formation and Persistent Group Formation. In Standard Group Formation two devices enter in a negotiation phase, in order to assign the role of the GO between them and to select the channel used bi the group. During the GO Negotiation the two devices send to each other a Negotiation request, this message includes a randomly chosen numeric value called "Intent value" and a tie breaking bit. The Intent value ranges from 0 to 15, and it measures the willingness of the device to assume the role of GO. There is not a standard definition of how to compute this value, but is left to the application layer. The tie breaking bit is randomly assigned by the first device sending the Negotiation request, while the responding device will reply with the tie breaking bit negated. The device with the higher Intent value will become GO, In case both devices Intent values sent are equal, the device with tie breaker bit set to 1 shall become GO. Finally thee device selected as GO will start a Group session, launching its soft-AP and the DHCP server. The other device can then connect to GO , typically via WiFi protected setup, and exchange data. At this point the group is established and also other devices can join the group as clients. In Autonomous Group Formation there is not a negotiation phase, the role is not chosen between some devices . Instead, a device announces itself as GO and starts a WD group sending beacons. This process is similar to the legacy Wi-Fi in which an AP directly sends beacons into the network to become discoverable. The Autonomous Group Formation is simpler and faster than Standard Group Formation. In Persistent Group Formation a device sends an invitation to another device, in order to reinstantiate a previously established WD group. The role of each device will remain the same as in the previous group. To establish a Persistent group, the devices must assign a flag bit inside the beacons during the formation of the group. If the flag is not set during Group Formation procedure, the P2P Devices cannot re-instantiate a Persistent group. The Wi-Fi Direct standard does not permit the Standard and Persistent Group Formation procedures between more than two devices. Other devices can only join, as clients, an already formed group. The solution is to use different methods to establish the GO, then wait for the selected GO to create the group in autonomous mode and finally let the other devices to connect to the formed group.

4.2 Group Formation

When exploiting WiFi Direct communications, the first fundamental phase is to establish a group of devices that participate. Then, it is crucial to elect an owner of the group. In our work, the GO is crucial also because is the responsible to deliver data of the group to the central collector. To form a group of peers and negotiate the role of the group owner, different methodologies can be taken into account. In [10], the authors define the most complex case as the standard case, and two simplified cases as the autonomous and persistent ones. In this work, I exploit the autonomous group formation for different reasons. First, it is simpler and faster than the standard case because there is no negotiation phase. Differently from the standard approach, a device announces itself as GO and starts a WD group sending beacons. Second, WD standard does not permit the standard and persistent Group Formation procedures between more than two devices. Other devices can only join, as clients, an already formed group. After having formed a group and elected an owner, other fundamental phases are the updates and the destruction of a group. I will explain them later in details according to different policies.

4.3 Group Formation Algorithm

This Section presents the proposed policies for group forming and group owner election. In this Section, I propose three different policies and I will explain how they can be effective according to different application scenarios. The first policy is called static grouping (SG) and it presents a static grid that covers the whole area of interest of the campaign. The second one is the point of interest-based grouping (PG) that set a place as the center of the group. The last one is the dynamic grouping (DG), which forms groups taking into account citizens staying close in a certain distance and usually moving.



(a) Static and POI (b) Dynamic (c) Dynamic

Figure 4.3. Different grouping methods

4.3.1 Static

This is the simplest GFA, it is based on the division of the area through a grid, where each quad has a sub-area of 50 m^2. The dimension is due to the reasonable distance of a WiFi Direct connection. In Fig. 4.8 we can see the city of Luxembourg divided for the Static GFA. To every quadrant in the grid will correspond a WiFi Direct Group. This GFA allows to cover the entire area of the MCS campaign. It is useful for task requiring a full coverage of the map, for instance a noise monitoring or air monitoring campaign. It allows easily to keep track if there will be certain zone with no contributors, and from this can born an user rewarding policy focused on solve the problem.

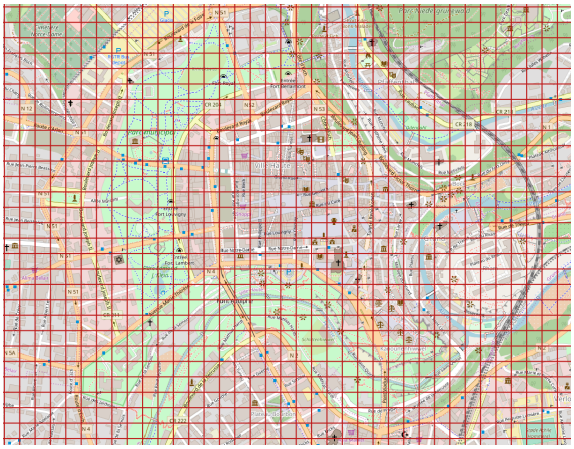


Figure 4.4. Luxembourg - grid division - Static GFA

Algorithm 3 Static GFA at device J

Initialize: Group(j)=0 ▷ Number of group of j, 0=NoGroup
GO(j)=false ▷ Group Owner Boolean flag
for $t = 1 \rightarrow T_j$ **do** ▷ For all the minutes of j's walk
  procedure INITIALIZATION
    Input: B_j, R_j, P_j^t, P_j^{t-1}, S_j ▷ Battery, RSSI, Position at t, Position at t-1, Speed
    Output: GI_j
    nq_j = findQuadrant(P_j^t) ▷ j's quadrant
    $\alpha_j^t \leftarrow$ bearing(P_j^t, P_j^{t-1}) ▷ Angle of direction from P_j^{t-1} to P_j^t
    $P_{exit} \leftarrow$ exitPoint(P_j^t, nq_j, α_j^t)
    $D = d(P_j^t, P_{exit})$ ▷ Linear distance from P_j^t to P_{exit}
    $T_e = D/S_j$ ▷ Estimation time to exit from nq_j
    $GI_j = W_t \cdot T_e + W_b \cdot B_j + W_r \cdot R_j$ ▷ W_t, W_b, W_r weights
  end procedure
  // Broadcast GI
  // $N_j \leftarrow$ wifiDiscovery ▷ List of j's neighbors
  procedure NEIGHBORHOOD DISCOVERY
    Input: N_j, nq_j
    Output: $Glist_j$ ▷ List of j's neighbors inside nq_j
    $Glist_j = \emptyset$
    for $m \in N_j$ **do**
      nq_m = findQuadrant(m) ▷ m's quadrant
      if $nq_m = nq_j$ **then**
        $Glist_j \leftarrow Glist_j \cup m$
      end if
    end for
  end procedure
  if Group(j)= nq_j **then**
    Continue
  else
    procedure GOELECTION
      GO(j)=false
      GI_{max} = firstElement($Glist_j$)
      if $GI_{max} = GI_j$ **then**
        GO(j)=true ▷ j becomes GO
        autonomousGroup() ▷ j creates the WD group
      else
        connect(ID(GI_{max})) ▷ j waits for the device with GI_{max} to connect
      end if
    end procedure
  end if
end for

Initialization Phase

Every single user has previously downloaded the map and the grid division of the area, therefore using it in combination with its own GPS sensor the user determines the number of the quad in which he is walking, saving the corresponding limit coordinates. Afterward, the user takes the coordinates of the actual position, those of the last registered position and he combines them together in order to find the angle of direction α, using as reference plane the true north. Fig. 4.5 shows an example of the direction determination process, where p_0 is the actual position and p_{-1} is the last registered one, the device every 60 seconds saves the pair *latitude* and *longitude* of its own position updating this variable, if during the calculation of the user's direction p_{-1} is equal to the initialization value (-1,-1) means that the user is participating to the campaign since less than 60 seconds, as there is no registered positions the direction is set to one of the default direction (North, East, South, West) depending on th closest limit of the quadrant.

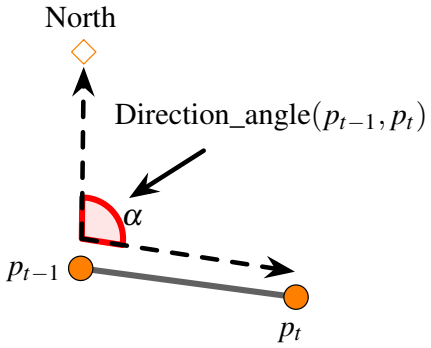


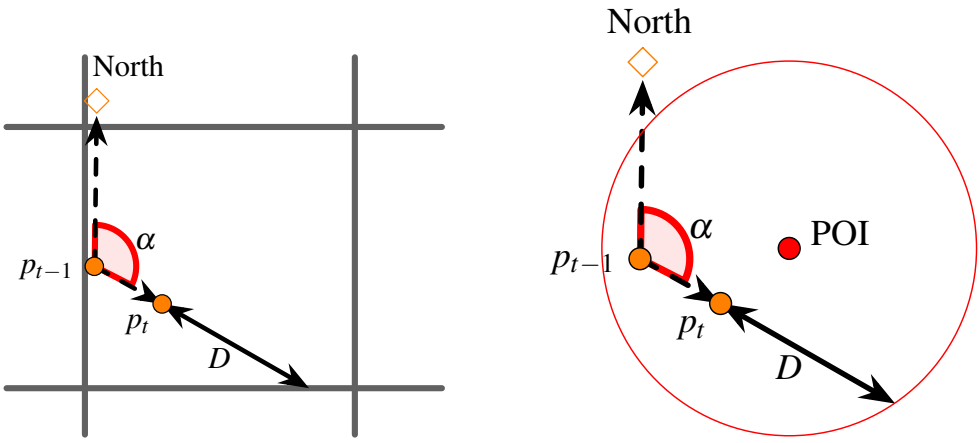
Figure 4.5. Angle of direction determination process

Next step is to compute the distance D between the actual position of the user and the external limit of his quadrant, assuming that the user will continue to walk along the same angle direction α. Fig. 4.6 shows an example of the determination process of this exit distance. After obtaining D, it is determined an estimation of the exit time T_e, which is the remaining time that the user spends inside the quadrant. T_e is computed by multiplying D with the actual speed of the user V_u. Finally, the user computes its own GOAI (GO Availability Index), that is the value representing the willingness of an user to assume the role of GO in the WiFi Direct

group. The GOAI is composed as following:

$$\text{GOAI} = W_t \cdot T_e + W_b \cdot Bat + W_r \cdot RSSI \tag{4.1}$$

where *Bat* denotes the remaining percentage of battery of the user's device, *RSSI* is the received signal strength indicator, W_t, W_b and W_r are the weights corresponding to the relevance inside the GOAI of the values that they multiply.



(a) Static (b) Poi

Figure 4.6. Exit distance determination process

Broadcast

Each device encodes its own GOAI into human-readable ASCII characters between [32,127] and insert it inside its own WiFi Device_name (DN), that is composed by:

$$DN = (NQ, ID, GOAI) \tag{4.2}$$

where *NQ* is the number of the quadrant and *ID* corresponds to the last 2 Bytes of the MAC address of the interface, so as to avoid *ID* collisions.

Neighborhood Discovery

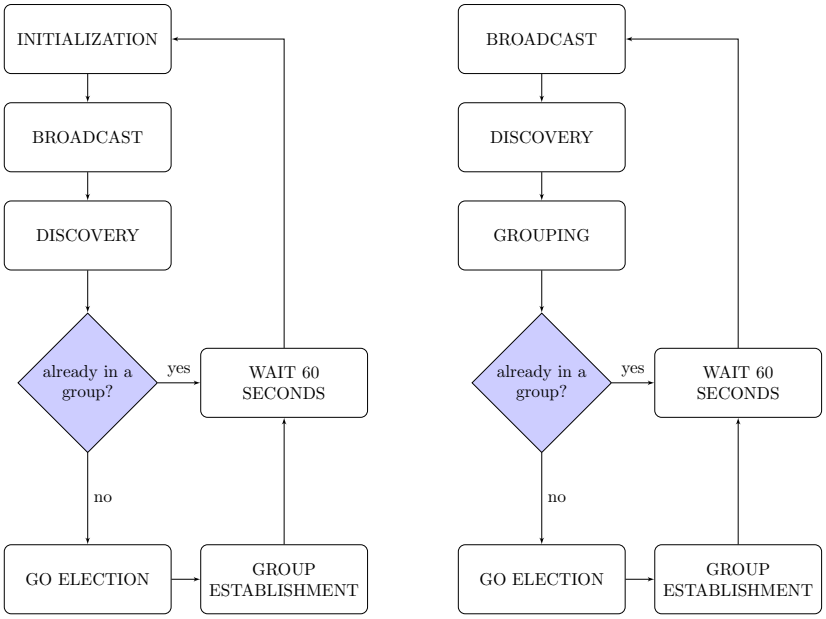
During this phase, a device collect other devices parameters, reading their name and saving into Glist only the *GOAI* of the nodes that have the same *NQ* of the device. The Glist will be sorted in descending order.

GO Election

Next, taking the first element of Glist, the device finds out the maximum *GOAI* inside the Glist and the corresponding *ID*. In case of equal *GOAI* is considered maximum the one with the biggest *ID*. if the ID corresponding to the maximum *GOAI* match the device's local *GOAI*, the device will start a WiFI Direct group in autonomous mode, and it will assume the role of GO. Otherwise, the device will simply wait for the elected ID to start the group and once that will happen, it will connect to the new GO.

Updating and new members

Making changes in a WiFi Direct group is unfeasible, once the GO is decided, if we want to change device that assumes that role, we will have to destroy the group and create a new one. To this end, if a new user arrives in a quadrant where a WiFi Direct group is already present, he will not create a new group but after the Discovery phase where he will generate his Glist, he will directly connect to the pre-existent group. In order to be prepared in case of disruption of the group, all members should update their *GOAI* every 60 seconds, repeating the initialization, the Broadcast and the Neighborhood Discovery phases. In the event that the actual GO exit from the quadrant, the group will be destroyed. All the members remained inside the quadrant will proceed simply with the GO election phase, this is possible thanks to the update phase that allows the devices to have always an up to date Glist.



(a) Static and POI (b) Dynamic

Figure 4.7. Phases Flowchart of different GFA

4.3.2 POI

The POI (Point Of Interest) GFA is very similar to the statistic one. What really changes is the application use, in deed it does not permit to cover the entire area but is focused on certain zones, defined by the position of specified type of POI. The development of this algorithm is due to the willingness to operate targeted MCS campaign, focusing only a certain category of areas. Fig. **??** shows an example of division of the city of Luxembourg, choosing the Bus stops as selected POI category.

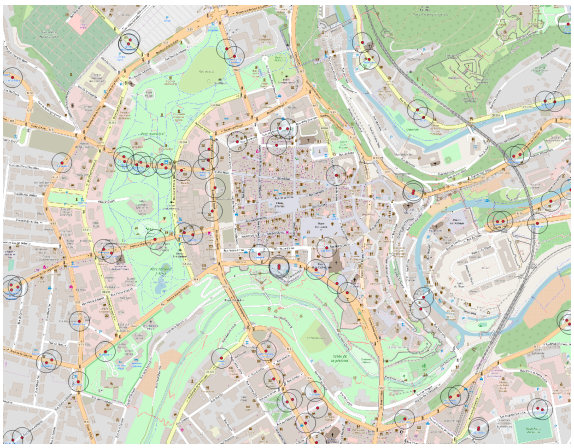


Figure 4.8. Luxembourg - Bus Stop POI division - POI based GFA

Differences with Static

Hence, the definition of the aggregation areas is not defined by a grid but through the position of the bus stops and a specific ray, within which a device standing there will be considered part of the group of that POI. The procedure is almost the same of the Statistic GFA, the only changes are in the Initialization phase. The device instead of determine in which quadrant it is, it will find out if it is inside the ray of a bus station. If the device will stand between more than one bus stations it will choose the closest one. Once the device has chosen the corresponding bus station, it will continue the normal Initialization phase but now *NQ* inside the device name will assume the value of the number of the assigned POI. The determination process of the distance from the outside of the area will not be calculated using the limits of the quadrant but instead will be used the borders of the circle determined by the ray around the POI. An example is shown in Fig. **??**

4.3.3 Dynamic

The dynamic GFA is designed to create group of devices going in the same direction. The motivation of this choice is due to the willingness to create stable group, limiting as much as possible destruction and recreation of groups, in order to create long lasting groups. The generated groups are not related to some fixed coordinates such as a grid or a POI. This GFA is based on the intent to establish

groups with a long connection keeping time (CKT).

Broadcast

First, each device publish its own data, the current position, the angle direction and the actual speed. In order to broadcast these data to other users, the values are inserted in a string used as device name field advertised to other nodes in proximity during the WiFI direct Discovery phase. The string looks like this

$$DN = (X, ID, LAT_t, LNG_t, LAT_{t-1}, LNG_{t-1}, V) \tag{4.3}$$

where X represents the actual status of the device, in this phase is set to B (Broadcast phase),ID corresponds to the last 2 Bytes of the MAC address of the interface, LAT_t,LNG_t are the coordinates of the current position, LAT_{t-1},LNG_{t-1} are the coordinates of the previous position, V is the actual speed of the user.

Discovery

In this phase, each device performs a simple WiFI direct device discovery, collecting all the device names of the nodes nearby. Between all the device names, we only take the ones that are in the status B and of these we save their parameters into a NodeList.

Grouping

In this phase we use the measurement of connection keeping time (CKT), it is an estimation of the time during which two devices will remain under a certain ray R, in this work I used 50 meters, due to the WiFi direct working distance. The computation of the CKT requires all the parameters provided by the devices: current position, last registered position and actual speed. For each element in NodeList, we compute CKT between the node in the list and the actual device. If the CKT is lower than the threshold value T_{th}, which varies according to the stability required from the MCS campaign, the corresponding node inside the Nodelist is deleted.
In order to define the GO, we introduce the concept of connectivity of a device, that

is described for each node as the number of neighbor devices around it, satisfying the *CKT* condition. Each Device will compute its value of the connectivity *C*, which will increase by one every time a neighbor of the device will satisfy the condition. *C* can be simply calculated counting the number of nodes inside Nodelist.Next the device computes its own GOAI, which is composed as following:

$$\text{GOAI} = W_c \cdot + W_b \cdot L_b + W_r \cdot RSSI \tag{4.4}$$

where L_b denotes the remaining percentage of battery of the user's device, *RSSI* is the received signal strength indicator, W_c, W_b and W_r are the weights corresponding to the relevance inside the GOAI of the values that they multiply. Afterwards the GOAI is published inside the device name and advertised to all neighbor, in this phase the status x will change to R (ready). Now the device name will look like this:

$$DN = (R, ID, GOAI) \tag{4.5}$$

Finally now the device performs an additional WiFi discovery, collecting all the device name of the neighbor and for each of them if the corresponding node is present inside the Nodelist, the device add the GOAI to a Glist, sorted in decreasing order.

GO election

In this phase the device finds the maximum GOAI from Glist. If the max GOAI corresponds to the one of the actual device, it changes its status in the device name to G (Group Owner) and it starts a new WiFi direct group in autonomous mode. Otherwise, it waits until the status of the node with the max GOAI change to G, consequentially the device change its status to m (member) and then connects to the group as soon as the GO creates it. If the node with the max GOAI changes its status to something different from G, the device deletes the corresponding node from the Nodelist and select the new maximum GOAI. Then with the new max GOAI, the device restarts the same previous procedure.

Update and group destruction

In order to keep the Glist and Nodelist of each device up to date, each device every 60 seconds has to repeat the Broadcast, Discovery and Grouping phase, also if the group is still standing. During an update, while repeating the Broadcast phase the status will not be set to B but will change to MB in case of members or GB in case of GO. While during the new Grouping phase the status instead of R will be MR for members and GR for GO . In case of destruction of the group for the GO disappearance or in case of a member moving away from his group, all the devices that will be without a group has simply to redo the GO election phase, this because they kept their Glist and Nodelist updated, thanks to the update procedure.

Fig. 4.15 shows the CDF of group duration during simulations. Dynamic groups have the longest duration, while static are for 90% less than 2 minutes.

4.4 Performance Evaluation

This Section at first introduces the evaluation scenario (Subsection 4.4.1) by explaining in detail the simulation setting, after provides details of the methodology to assess the energy consumption(Subsection 4.4.2) and then presents the results (Subsection 4.4.3).

4.4.1 Simulation Setting

For performance evaluation, I run experiments in the city center of Luxembourg exploiting CrowdSenSim. I compare individual and collaborative data collection and different collaborative approaches.
As explained in Section 3.2, 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. 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

Algorithm 4 Dynamic GFA at device J

Initialize: GO(j)=false ▷ Group Owner Boolean flag
for $t = 1 \rightarrow T_j$ **do** ▷ For all the minutes of j's walk
  procedure INITIALIZATION
    Input: B_j, R_j, P_j^t, P_j^{t-1}, S_j ▷ Battery, RSSI, Position at t, Position at t-1, Speed
    $status(j) \leftarrow B$ ▷ Status of J
    Broadcast(P_j^t, P_j^{t-1},S_j)
  end procedure
  // $N_j \leftarrow$ wifiDiscovery ▷ List of j's neighbors
  procedure NEIGHBORHOOD DISCOVERY
    Input: N_j
    Output: $Nlist_j$ ▷ List of j's neighbors in B state
    $Nlist_j = \emptyset$
    for $n \in Nlist_j$ **do**
      if $status(n) = B$ **then**
        $Nlist_j \leftarrow Nlist_j \cup n$
      end if
    end for
  end procedure
  procedure GROUPING
    Input: $Nlist_j$, j
    Output: GI_j,$Nlist_j$ ▷ Go index of j, list of j's neighbors with CKT under T_{th}
    $C = 0$
    for $n \in Nlist_j$ **do**
      CKT_{jn} =computeCKT(j,n)
      if $CKT_{jn} > T_{th}$ **then**
        Delete n
      else
        $C = C + 1$
      end if
    end for
    $GI_j = W_c \cdot C + W_b \cdot B_j + W_r \cdot R_j$ ▷ W_c, W_b, W_r weights
    $status(j) \leftarrow R$
    Broadcast(GI_j)
  end procedure
  $Glist_j = \emptyset$
  // $Glist_j \leftarrow$ wifiDiscovery
  procedure GOELECTION
    GO(j)=false
    for $g \in Glist_j$ **do**
      GI_{max} = firstElement($Glist_j$)
      if $GI_{max} = GI_j$ **then**
        GO(j)=true ▷ j becomes GO
        autonomousGroup() ▷ j creates the WD group
      else
        if GI_{max}) = $true$ **then**
          connect(ID(GI_{max})) ▷ j waits for the device with GI_{max} to connect
        else
          continue
        end if
      end if
    end for
  end procedure
end for

distributed between [20, 40] minutes. 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 set to 50 000. 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). The initial battery level when users start walking is uniformly distributed in the range [10-90]%.

4.4.2 Proposed methodology for energy profiling

ù The energy consumption model takes separately into account both costs associated to reporting and sensing [13]. I denote with E_s^c the energy consumption attributed to sensing and with E_s^r the energy cost associated to reporting for sensor s. Then:

$$E_s = E_s^c + E_s^r. \tag{4.6}$$

The contribution E_s^c due to sensing operation is considered only if sensor s is not already in use for personal usage or another application. Table 4.2a shows the sensing equipment exploited for the simulations and their corresponding parameters. Data reporting implies sending data generated from the set of sensors $\mathcal{S}$ to the central collector using available communication technologies. Data transmission is always executed at the beginning of the timeslot t for samples generated during timeslot $t - 1$.

To compute the power consumption of communication equipment, I refer to[34]. I take into account the WiFi Direct power consumption for uplink and downlink communications because devices exchange data, while only the uplink power consumption of LTE because I exploit it for devices reporting data to the cloud collector, that are users assuming roles of group owner or no group. Assuming uplink throughput is T_u and downlink throughput is T_d, the instant WiFi Direct

power level for uplink is

$$P_u^W = \alpha_u^W \cdot T_u^W + \beta^W \tag{4.7}$$

and for downlink is

$$P_d^W = \alpha_d \cdot T_d^W + \beta^W \tag{4.8}$$

The LTE power level for uplink is

$$P_u^L = \alpha_u^L \cdot T_u^L + \beta^L \tag{4.9}$$

The best fit α and β parameters are listed in Table 4.2b, the throughput T for WifiDirect depends on the inter device distance and follows the curves described in fig.4.9 that is taken from [51], while throughput for LTE communications depends on curves taken from [34].

The interface used for the reporting depends on the role assumed by the user in the timeslot t.

- Members transmit the sensing data to the group owner using only the WiFi interface.
- Group owners use the WiFi interface to download data from members and then upload it to the collector through LTE interface.
- No group users do not report data

The energy consumption spent for transmission E^{tx} relative to the transmission time τ_{tx} is defined as:

$$E^t x = \int_0^{\tau_{tx}} P_{tx} \, \mathrm{d}t, \tag{4.10}$$

where P_{tx} is the power consumed for transmissions relative to the communication interface used by the user.

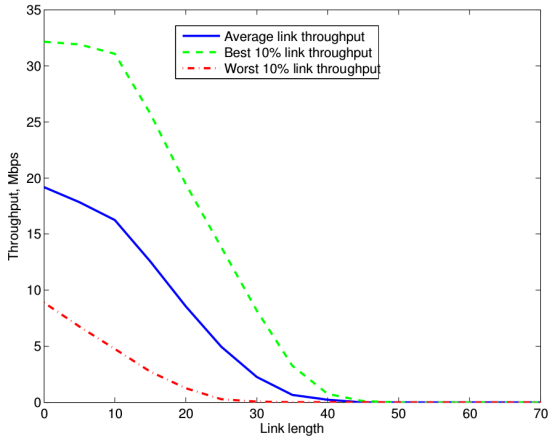


Figure 4.9. Best 10%, worst 10% and average WiFi direct throughput.

Table 4.1. Sensor and communication equipment parameters used for performance evaluation

Sensor	Parameter	Value	Unit
Accelerometer	Sample rate	4	kHz
	Sample size	6	Bytes
	Current	450	µA
Proximity	Sample rate	8.1	MHz
	Sample size	2	Bytes
	Current	150	µA
GPS	Update period	10	s
	Sample size	24	Bytes
	Current	23	mA

(a) Sensor Equipment

	Symbol	Value	Unit
LTE	α_u^L	438.39	mW/Mbps
	β^L	1288.04	mW
WiFi Direct	α_u^W	283.17	mW/Mbps
	α_d^W	137.01	mW/Mbps
	β^W	132.86	mW

(b) Communication Equipment

4.4.3 Simulation Results

For performance evaluation, I attend experiments to assess the different characteristics of each grouping strategies. In particular i analyzed the energy consumption, the amount of collected data and other aspect of each group, such as duration or number of members.

Energy Consumption. Fig. 4.10 show the CDF of energy consumption for reporting data to the cloud collector with different approaches. The CDF compares the three grouping strategies with the consumption of an individual campaign, where users send data to collector autonomously through LTE interface.

Interestingly, the difference between individual and collaborative approach is substantial. Collaborative approaches spend less energy for approximately 70% of users. The reason is that even though with collaborative there is an additional quantity of energy caused by the WiFi Direct, the energy consumption spent by members in a group is significantly less respect the one of a normal user in individual approach. An other meaningful aspect is that in collaborative there is a better selection of useful participants, as mentioned before in all the methodologies there is a limit of contributing users, beyond which other users will not participate to the campaign because are not needed.
To understand the energy consumption and the distinctions between the approaches, the percentage roles of users (Fig. 4.11) becomes crucial. As expected, Dynamic is the most energy consuming approach, it allows the biggest percentage of group owner, while for the other strategies there is a fixed maximum number of possible groups due to the grid or the poi in dynamic the maximum number of groups is related only with the number of users involved. The ratio among the quantity of group owners and members is one third, while the percentage of no group users is less than 25%. The consumption with statistic approach is not so lower, this because the fixed number of groups is very high, in the area that I simulate there are 700 grid of 50m^2. Moreover, the only no group users in a campaign based on this approach derive from the excess of users on a specific grid, due to the full coverage of the area there are not users walking alone and not inserted in any group. Indeed the percentage of no groups is 50%. The poi is

the most focused strategy, it has the lowest energy consumption and the lowest participants to the campaign, due to the low coverage of the campaign area, the percentage of no group users is close to 90%, while the ratio go/memebers is one sixth.

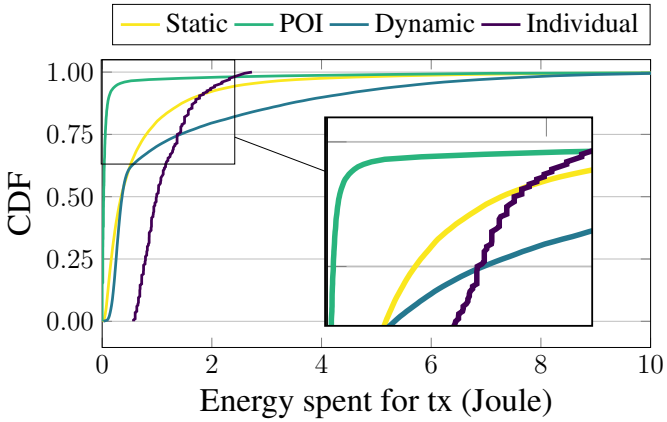


Figure 4.10. CDF of energy consumption.

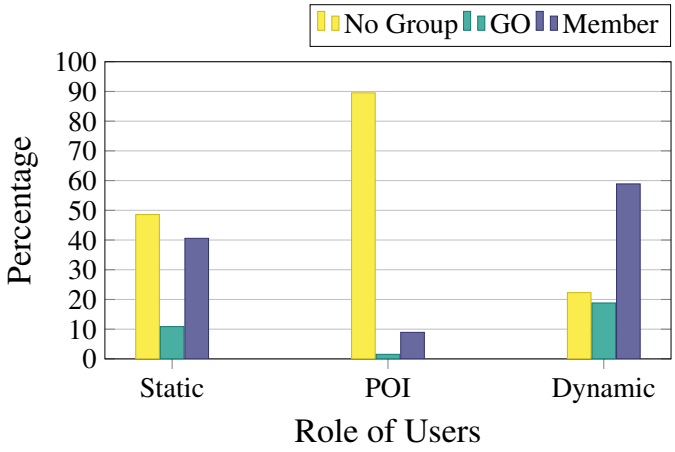


Figure 4.11. Percentage of roles assumed by the users during campaigns based on different GFA's.

As mentioned before, collaborative framework enables the majority of users to spent less than in individual campaigns, fig. 4.13 shows the roles assumed by users that spent more energy than the individual users and the average number of members in case they became group owner during their participation, this figure highlights that the percentage of group owners is clearly higher respect all users and the average number of members in case of group owners is very close to the

maximum value 10, except that for dynamic where this average is 4, the reason is that in this approach we have many group owners and less members per group, indeed the percentage of group owners is 75%.

On the other hand, fig. 4.12 shows the same aspects of fig. 4.13 but referring to users that spent less than users in individual. It is clear how in this case for all the grouping strategies we have an higher percentage of members and no group users, while the average number of members is always lower than 3.

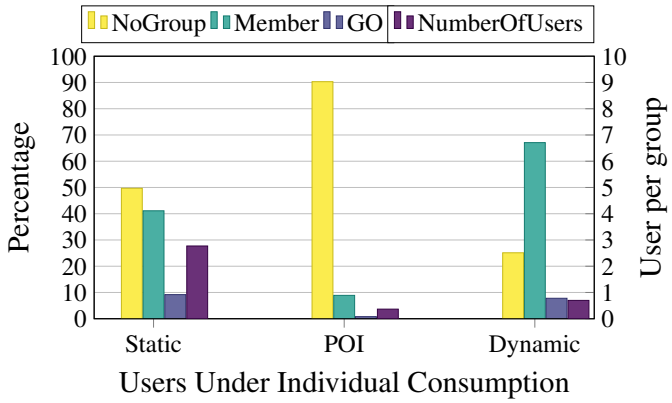


Figure 4.12. Roles assumed by users that spent less than in individual approach

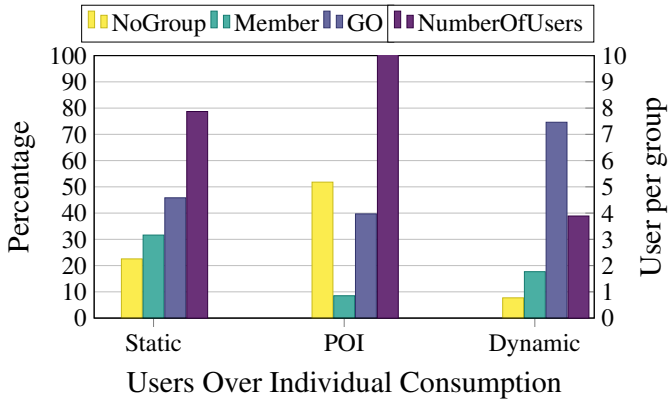


Figure 4.13. Roles assumed by users that spent more than in individual approach

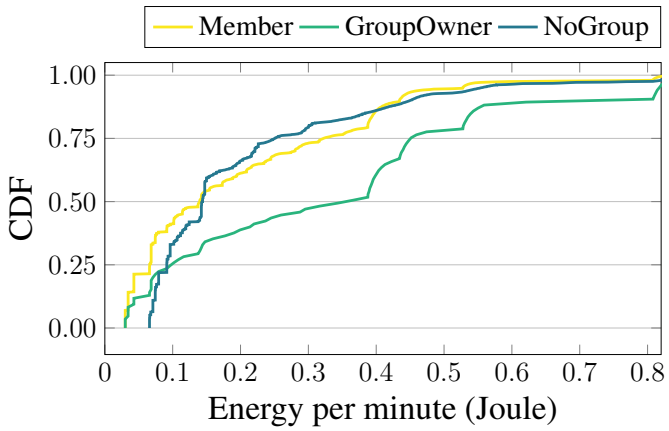
Fig.4.14 illustrates the comparison of per minute consumption per different roles for each grouping strategies. At first, fig.4.14a shows that the group owners in poi approach report consumption is much greater than other two approaches. The reason of this is that poi grouping is characterized by an higher number of

members, I will details it later with Fig.4.17.

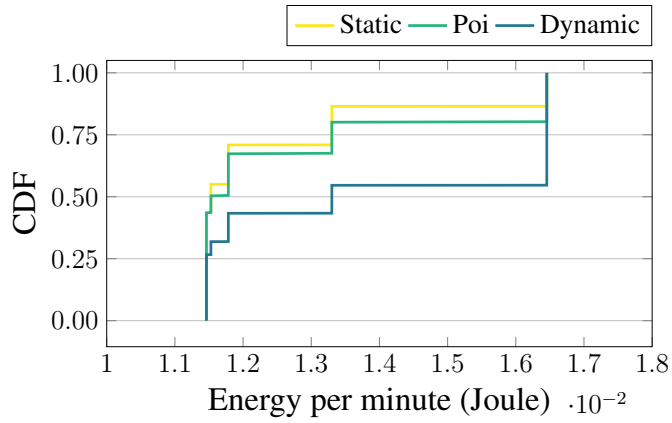
Fig.4.14b compares the per minute consumption of members, here the differences are related only with the inter device distance that determines the power consumption for the WiFi Direct. It is possible to note that in dynamic we have longest inter device distances, this because most of the time in poi and static the go is selected in the middle of the grid or close to the poi, in this way the average inter distance will be half WiFi Direct range 25m, while in dynamic do not exist any restrictions and groups can live until two users are closer than the WiFi Direct range 50m.

As expected, Fig.4.14c shows that are not any differences between strategies for the consumption of no group users.

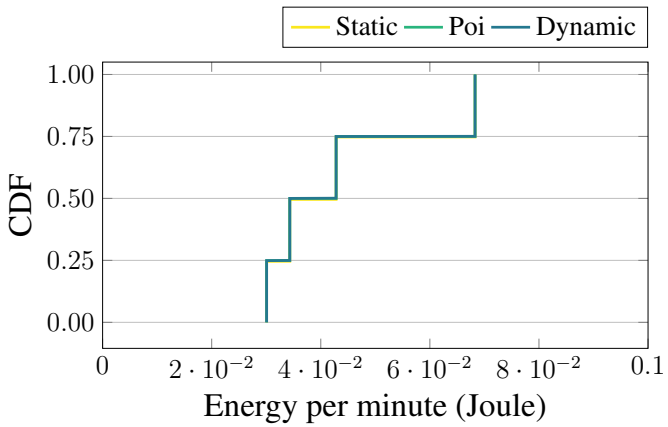
Amount of Collected Data. Fig. 4.16 exhibits the total amount of data generated from users with different grouping approaches during their existences. It shows that static approach generated the lowest amount of data. The 75% of groups in poi and static approaches generate the same amount of data, the reason of that can be attributed to two different motivations: (i) the different times of existence of groups, , as fig. 4.15 shows the 75% of groups in static and poi exist for 1 minute, while dynamic according to its purpose creates longer standing groups, (ii) the number of members per group in each timeslot, fig. 4.17 illustrates the CDF of number of members per group, dynamic provides the lowest number of members, while in poi and static respectively 50% and 25% of timeslots reach the maximum number of members.



(a) Group owner



(b) Member



(c) No group

Figure 4.14. Energy per minute spent for transmission for each role of users

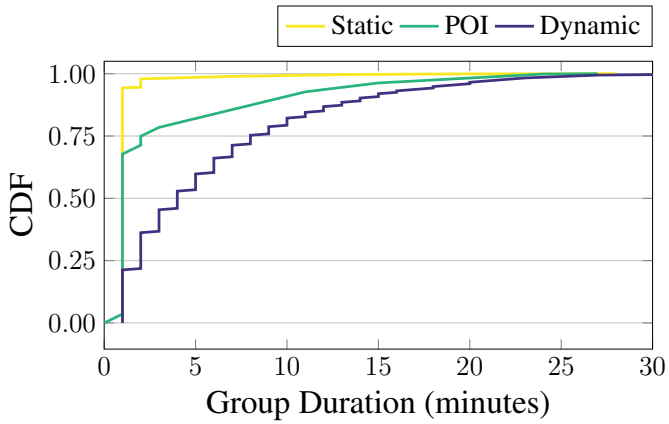


Figure 4.15. CDF of Group Duration.

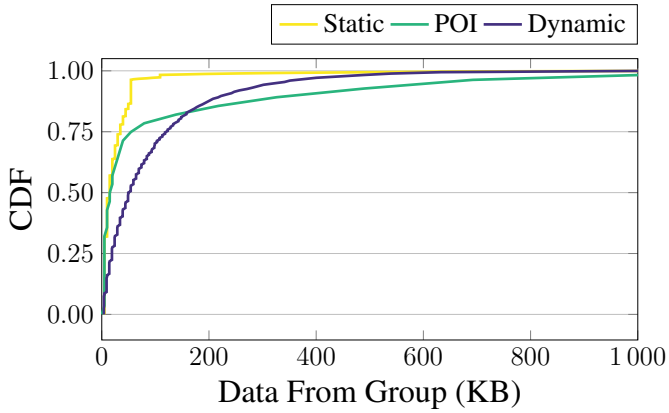


Figure 4.16. CDF of Data Collected by groups during their life.

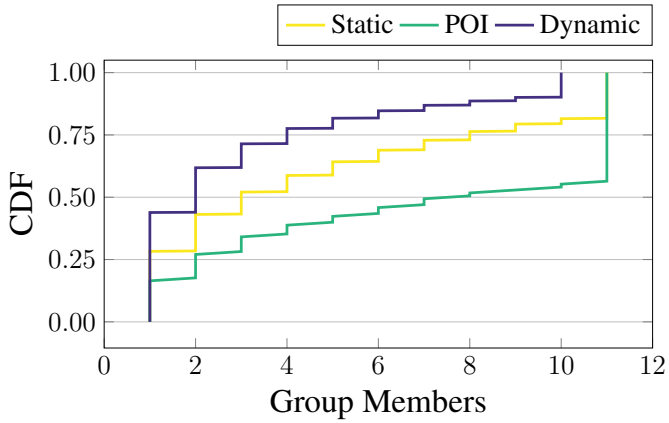


Figure 4.17. CDF of number of members per group.

Chapter 5

Conclusion and Future Work

This thesis proposes a novel collaborative framework for task allocation and data reporting in MCS systems. The framework aims at minimizing the energy consumption for the participants in reporting while allocating tasks to a group of peers instead of an individual. I proposed and compared three different strategies to form groups and perform owner election.

The evaluation and the analysis of the collaborative framework is evaluated through simulations in a real urban environment with a large number of participants. To perform simulations I used CrowdSenSim , a discrete-event simulator for mobile crowdsensing developed at University of Luxembourg. The simulation results confirm the convenience of the proposed framework, Collaborative approaches spend less energy for approximately 70% of users. Moreover, the energy consumption spent by members in a group is significantly less respect the one of a normal user in individual approach.

Using CrowdSenSim to perform extensive analysis, I investigated all the weak points and the issue related to the simulator. In order to fill these gaps, I extensively studied models for pedestrian mobility for smart city simulators. First, I applied a procedure, AOP, to augment the precision of the street network provided by OpenStreetMap. The optimal version relies on the Vicenty's formula to determine the distance between vertices in the graph. The linearized version approximates this distance exploiting the Euclidean formula. The resulting graph is suitable for inclusion in CrowdSenSim, a simulator for MCS. I implemented two different

user arrival models, U-MOB and D-MOB. The latter is based on a well-known real data set, ParticipAct. For performance evaluation, we assessed the scalability of AOP, the accuracy of the mobility models and verified popular metrics such as the average per-user number of contacts and the stability of the contacts. Interestingly, the U-MOB mobility model approximates very well D-MOB. Thus, for simulations with large population, uniform user arrivals provide an efficient method to obtain realistic simulation for human mobility.

To extend current functionalities of the collaborative framework, as future work it is possible to consider the development of an adaptive framework that enables different device to device communication, such as LTE direct or Bluetooth, changing communication technology basing on the purpose of the task and the consumption of the users.

However, future development for CrowdSenSim are twofold. First, the implementation of more accurate communication model to analyze in more details the networking aspects of MCS systems. Second, the development of a mobility where researchers can define directions of user movements on individual basis.

Bibliography

[1] A. Ahmed, K. Yasumoto, Y. Yamauchi, and M. Ito. "Distance and time based node selection for probabilistic coverage in People-Centric Sensing". In: *2011 8th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*. 2011, pp. 134–142. DOI: 10.1109/SAHCN.2011.5984884.

[2] A. Asadi and V. Mancuso. "Network-Assisted Outband D2D-Clustering in 5G Cellular Networks: Theory and Practice". In: *IEEE Transactions on Mobile Computing* 16.8 (2017), pp. 2246–2259. ISSN: 1536-1233. DOI: 10.1109/TMC.2016.2621041.

[3] Jesus M Barajas, Geoff Boeing, and Julie Wartell. "Neighborhood Change, One Pint at a Time: The Impact of Local Characteristics on Craft Breweries". In: *Untapped: Exploring the Cultural Dimensions of Craft Beer*. West Virginia University Press, 2017, pp. 155–176.

[4] Michael Behrisch, Laura Bieker, Jakob Erdmann, and Daniel Krajzewicz. "SUMO–simulation of urban mobility: an overview". In: *SIMUL*. ThinkMind. 2011.

[5] Jacopo De Benedetto, Paolo Bellavista, and Luca Foschini. "Proximity discovery and data dissemination for mobile crowd sensing using LTE direct". In: *Computer Networks* 129 (2017). Special Issue on 5G Wireless Networks for IoT and Body Sensors, pp. 510 –521. ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2017.08.002. URL: http://www.sciencedirect.com/science/article/pii/S1389128617303092.

[6] 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.

[7] D. Bonino, M. T. D. Alizo, C. Pastrone, and M. Spirito. "WasteApp: Smarter waste recycling for smart citizens". In: *International Multidisciplinary Conference on Computer and Energy Science (SpliTech)*. 2016, pp. 1–6. DOI: 10.1109/SpliTech.2016.7555951.

[8] M. R. Brust, M. Zurad, L. Hentges, L. Gomes, G. Danoy, and P. Bouvry. "Target Tracking Optimization of UAV Swarms Based on Dual-Pheromone Clustering". In: *3rd IEEE International Conference on Cybernetics (CYBCONF)*. 2017, pp. 1–8.

[9] D. Camps-Mur, A. Garcia-Saavedra, and P. Serrano. "Device-to-device communications with Wi-Fi Direct: overview and experimentation". In: *IEEE Wireless Communications* 20.3 (2013), pp. 96–104. ISSN: 1536-1284. DOI: 10.1109/MWC.2013.6549288.

[10] D. Camps-Mur, A. Garcia-Saavedra, and P. Serrano. "Device-to-device communications with Wi-Fi Direct: overview and experimentation". In: *IEEE Wireless Communications* 20.3 (2013), pp. 96–104. ISSN: 1536-1284. DOI: 10.1109/MWC.2013.6549288.

[11] A. Capponi, C. Fiandrino, C. Franck, U. Sorger, D. Kliazovich, and P. Bouvry. "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)*. 2016, pp. 456–459. DOI: 10.1109/CloudCom.2016.0077.

[12] A. Capponi, C. Fiandrino, D. Kliazovich, P. Bouvry, and S. Giordano. "Energy efficient data collection in opportunistic mobile crowdsensing architectures for smart cities". In: *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. 2017, pp. 307–312. DOI: 10.1109/INFCOMW.2017.8116394.

[13] Andrea Capponi, Claudio Fiandrino, Dzmitry Kliazovich, Pascal Bouvry, and Stefano Giordano. "A Cost-Effective Distributed Framework for Data Collection in Cloud-based Mobile Crowd Sensing Architectures". In: *IEEE Transactions on Sustainable Computing* (2017), pp. 1–1.

[14] C. E. Casetti, C. F. Chiasserini, Y. Duan, P. Giaccone, and A. Perez Manriquez. "Data Connectivity and Smart Group Formation in Wi-Fi Direct Multi-Group Networks". In: *IEEE Transactions on Network and Service Management* 15.1 (2018), pp. 245–259. ISSN: 1932-4537. DOI: 10.1109/TNSM.2017.2766124.

[15] A. Chaintreau, P. Hui, J. Crowcroft, C. Diot, R. Gass, and J. Scott. "Impact of Human Mobility on Opportunistic Forwarding Algorithms". In: *IEEE Transactions on Mobile Computing* 6.6 (2007), pp. 606–620. ISSN: 1536-1233. DOI: 10.1109/TMC.2007.1060.

[16] A. Chatterjee, M. Borokhovich, L. R. Varshney, and S. Vishwanath. "Efficient and flexible crowdsourcing of specialized tasks with precedence constraints". In: *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*. 2016, pp. 1–9. DOI: 10.1109/INFOCOM.2016.7524615.

[17] Qinghua Chen, Zhengqiu Weng, Yang Han, and Yanmin Zhu. "A distributed algorithm for maximizing utility of data collection in a crowd sensing system". In: *International Journal of Distributed Sensor Networks* 12.9 (2016), p. 1550147716668083. DOI: 10.1177/1550147716668083. eprint: https://doi.org/10.1177/1550147716668083. URL: https://doi.org/10.1177/1550147716668083.

[18] Stefano Chessa, Michele Girolami, Luca Foschini, Raffaele Ianniello, Antonio Corradi, and Paolo Bellavista. "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.

[19] M. Conti, F. Delmastro, G. Minutiello, and R. Paris. "Experimenting opportunistic networks with WiFi Direct". In: *2013 IFIP Wireless Days (WD)*. 2013, pp. 1–6. DOI: 10.1109/WD.2013.6686501.

[21] Matthew L Daggitt, Anastasios Noulas, Blake Shaw, and Cecilia Mascolo. "Tracking urban activity growth globally with big location data". In: *Open Science* 3.4 (2016).

[22] K. Farkas and I. Lendák. "Simulation environment for investigating crowd-sensing based urban parking". In: *International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*. 2015, pp. 320–327. DOI: 10.1109/MTITS.2015.7223274.

[23] C. Fiandrino, A. Capponi, G. Cacciatore, D. Kliazovich, U. Sorger, P. Bouvry, B. Kantarci, F. Granelli, and S. Giordano. "CrowdSenSim: a Simulation Platform for Mobile Crowdsensing in Realistic Urban Environments". In: *IEEE Access* 5 (2017), pp. 3490–3503. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2017.2671678.

[24] C. Fiandrino, A. Capponi, G. Cacciatore, D. Kliazovich, U. Sorger, P. Bouvry, B. Kantarci, F. Granelli, and S. Giordano. "CrowdSenSim: a Simulation Platform for Mobile Crowdsensing in Realistic Urban Environments". In: *IEEE Access* 5 (2017), pp. 3490–3503. DOI: 10.1109/ACCESS.2017.2671678.

[25] C. Fiandrino, B. Kantarci, F. Anjomshoa, D. Kliazovich, P. Bouvry, and J. Matthews. "Sociability-Driven User Recruitment in Mobile Crowdsensing Internet of Things Platforms". In: *IEEE Global Communications Conference (GLOBECOM)*. 2016, pp. 1–6. DOI: 10.1109/GLOCOM.2016.7842272.

[26] Holger Füßler, Jörg Widmer, Michael KÃ€semann, Martin Mauve, and Hannes Hartenstein. "Contention-based forwarding for mobile ad hoc networks". In: *Ad Hoc Networks* 1.4 (2003), pp. 351 –369. ISSN: 1570-8705. DOI: https://doi.org/10.1016/S1570-8705(03)00038-6. URL: http://www.sciencedirect.com/science/article/pii/S1570870503000386.

[27] R. K. Ganti, F. Ye, and H. Lei. "Mobile crowdsensing: current state and future challenges". In: *IEEE Communications Magazine* 49.11 (2011), pp. 32–39. ISSN: 0163-6804. DOI: 10.1109/MCOM.2011.6069707.

[28] Michael F. Goodchild. "Citizens as sensors: the world of volunteered geography". In: *GeoJournal* 69.4 (2007), pp. 211–221. ISSN: 1572-9893. DOI:

10.1007/s10708-007-9111-y. URL: http://dx.doi.org/10.1007/s10708-007-9111-y.

[29] B. Guo, Y. Liu, W. Wu, Z. Yu, and Q. Han. "ActiveCrowd: A Framework for Optimized Multitask Allocation in Mobile Crowdsensing Systems". In: *IEEE Transactions on Human-Machine Systems* 47.3 (2017), pp. 392–403. ISSN: 2168-2291. DOI: 10.1109/THMS.2016.2599489.

[30] S. Hachem, A. Pathak, and V. Issarny. "Probabilistic registration for large-scale mobile participatory sensing". In: *2013 IEEE International Conference on Pervasive Computing and Communications (PerCom)*. 2013, pp. 132–140. DOI: 10.1109/PerCom.2013.6526723.

[31] 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 (2016), pp. 1462–1475. ISSN: 1063-6692. DOI: 10.1109/TNET.2015.2418191.

[32] David Hasenfratz, Olga Saukh, Silvan Sturzenegger, and Lothar Thiele. "Participatory Air Pollution Monitoring Using Smartphones". In: *In Mobile Sensing: From Smartphones and Wearables to Big Data*. Beijing, China: ACM, 2012.

[33] S. He, D. H. Shin, J. Zhang, and J. Chen. "Toward optimal allocation of location dependent tasks in crowdsensing". In: *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*. 2014, pp. 745–753. DOI: 10.1109/INFOCOM.2014.6848001.

[34] Junxian Huang, Feng Qian, Alexandre Gerber, Z Morley Mao, Subhabrata Sen, and Oliver Spatscheck. "A close examination of performance and power characteristics of 4G LTE networks". In: *10th ACM International conference on Mobile systems, applications, and services*. 2012, pp. 225–238.

[36] K. Jahed, O. Farhat, G. Al-Jurdi, and S. Sharafeddine. "Optimized group owner selection in WiFi direct networks". In: *2016 24th International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*. 2016, pp. 1–5. DOI: 10.1109/SOFTCOM.2016.7772169.

[37] Luis G Jaimes, Idalides Vergara-Laurens, and Miguel A Labrador. "A location-based incentive mechanism for participatory sensing systems with budget constraints". In: *Pervasive Computing and Communications (PerCom), 2012 IEEE International Conference on*. IEEE. 2012, pp. 103–108.

[38] 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).

[39] Pablo Jensen. "A network-based prediction of retail stores commercial categories and optimal locations". In: *Physical Review E : Statistical, Nonlinear, and Soft Matter Physics* 74 (2006).

[40] Sunyoung Kim, Christine Robson, Thomas Zimmerman, Jeffrey Pierce, and Eben M. Haber. "Creek Watch: Pairing Usefulness and Usability for Successful Citizen Science". In: *SIGCHI Conference on Human Factors in Computing Systems*. CHI. Vancouver, BC, Canada: ACM, 2011. ISBN: 978-1-4503-0228-9. DOI: 10.1145/1978942.1979251.

[41] Nicholas D. Lane, Yohan Chon, Lin Zhou, Yongzhe Zhang, Fan Li, Dongwon Kim, Guanzhong Ding, Feng Zhao, and Hojung Cha. "Piggyback Crowd-Sensing (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.

[42] Nicholas D. Lane, Yohan Chon, Lin Zhou, Yongzhe Zhang, Fan Li, Dongwon Kim, Guanzhong Ding, Feng Zhao, and Hojung Cha. "Piggyback Crowd-Sensing (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.

[43] C.H. Liu, B. Zhang, X. Su, J. Ma, W. Wang, and K.K. Leung. "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.

[44] Hong Lu, Nicholas D. Lane, Shane B. Eisenman, and Andrew T. Campbell. "Bubble-sensing: Binding sensing tasks to the physical world". In: *Pervasive and Mobile Computing* 6.1 (2010), pp. 58 –71. ISSN: 1574-1192. DOI: https://doi.org/10.1016/j.pmcj.2009.10.005. URL: http://www.sciencedirect.com/science/article/pii/S157411920900100X.

[45] Kamal Mehdi, Massinissa Lounis, Ahcène Bounceur, and Tahar Kechadi. "CupCarbon: A Multi-agent and Discrete Event Wireless Sensor Network Design and Simulation Tool". In: *7th International ICST Conference on Simulation Tools and Techniques*. SIMUTools '14. Lisbon, Portugal, 2014, pp. 126–131. ISBN: 978-1-63190-007-5.

[46] 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.

[47] Pascal Neis, Dennis Zielstra, and Alexander Zipf. "The Street Network Evolution of Crowdsourced Maps: OpenStreetMap in Germany 2007–2011". In: *Future Internet* 4.1 (2012), pp. 1–21. ISSN: 1999-5903. DOI: 10.3390/fi4010001.

[48] Daniel Peebles, Hong Lu, Nicholas D. Lane, Tanzeem Choudhury, and Andrew T. Campbell. "Community-Guided Learning: Exploiting Mobile Sensor Users to Model Human Behavior". In: *AAAI*. 2010.

[49] M. Pouryazdan, C. Fiandrino, B. Kantarci, D. Kliazovich, T. Soyata, and P. Bouvry. "Game-Theoretic Recruitment of Sensing Service Providers for Trustworthy Cloud-Centric Internet-of-Things (IoT) Applications". In: *IEEE Globecom Workshops (GC Wkshps)*. 2016, pp. 1–6. DOI: 10.1109/GLOCOMW.2016.7848915.

[50] M. Pouryazdan, C. Fiandrino, B. Kantarci, D. Kliazovich, T. Soyata, and P. Bouvry. "Game-Theoretic Recruitment of Sensing Service Providers for Trustworthy Cloud-Centric Internet-of-Things (IoT) Applications". In: *IEEE*

Globecom Workshops (GC Wkshps). 2016, pp. 1–6. DOI: 10.1109/GLOCOMW.2016.7848915.

[51] A. Pyattaev, K. Johnsson, S. Andreev, and Y. Koucheryavy. "3GPP LTE traffic offloading onto WiFi Direct". In: *2013 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*. 2013, pp. 135–140. DOI: 10.1109/WCNCW.2013.6533328.

[52] C. Ragona, F. Granelli, C. Fiandrino, D. Kliazovich, and P. Bouvry. "Energy-Efficient Computation Offloading for Wearable Devices and Smartphones in Mobile Cloud Computing". In: *IEEE Global Communications Conference (GLOBECOM)*. 2015, pp. 1–6. DOI: 10.1109/GLOCOM.2015.7417039.

[53] Rajiv Ranjan, Meisong Wang, Charith Perera, Prem Prakash Jayaraman, Miranda Zhang, Peter Strazdins, and R.K. Shyamsundar. "City Data Fusion: Sensor Data Fusion in the Internet of Things". In: *Int. J. Distrib. Syst. Technol.* 7.1 (2016), pp. 15–36. ISSN: 1947-3532. DOI: 10.4018/IJDST.2016010102.

[54] Sasank Reddy, Deborah Estrin, and Mani Srivastava. "Recruitment Framework for Participatory Sensing Data Collections". In: *Proceedings of the 8th International Conference on Pervasive Computing*. Pervasive'10. Helsinki, Finland: Springer-Verlag, 2010, pp. 138–155. ISBN: 3-642-12653-7, 978-3-642-12653-6. DOI: 10.1007/978-3-642-12654-3_9. URL: http://dx.doi.org/10.1007/978-3-642-12654-3_9.

[55] Sasank Reddy and Vids Samanta. *Urban Sensing: Garbage Watch*. UCLA Center for Embedded Networked Sensing. 2011.

[56] Sasank Reddy, Katie Shilton, Gleb Denisov, Christian Cenizal, Deborah Estrin, and Mani B. Srivastava. "Biketastic: sensing and mapping for better biking". In: *CHI*. 2010.

[57] CC Robusto. "The cosine-haversine formula". In: *The American Mathematical Monthly* 64.1 (1957), pp. 38–40.

[58] João G. P. Rodrigues, Ana Aguiar, and João Barros. "SenseMyCity: Crowdsourcing an Urban Sensor". In: *CoRR* abs/1412.2070 (2014).

[59] Immanuel Schweizer, Roman Bärtl, Axel Schulz, Florian Probst, and Max Mühläuser. "Noisemap - real-time participatory noise maps". In: *In Second International Workshop on Sensing Applications on Mobile Phones*. 2011, pp. 1–5.

[60] A. Sciarrone, C. Fiandrino, I. Bisio, F. Lavagetto, D. Kliazovich, and P. Bouvry. "Smart Probabilistic Fingerprinting for Indoor Localization over Fog Computing Platforms". In: *5th IEEE International Conference on Cloud Networking (CloudNet)*. 2016, pp. 39–44. DOI: 10.1109/CloudNet.2016.43.

[61] Xiang Sheng, J. Tang, and W. Zhang. "Energy-efficient collaborative sensing with mobile phones". In: *2012 Proceedings IEEE INFOCOM*. 2012, pp. 1916–1924. DOI: 10.1109/INFCOM.2012.6195568.

[62] V. Sivaraman, J. Carrapetta, K. Hu, and B. G. Luxan. "HazeWatch: A participatory sensor system for monitoring air pollution in Sydney". In: *IEEE LCN*. 2013. DOI: 10.1109/LCNW.2013.6758498.

[63] T. J. Steiner, G. Huang, and J. J. Leonard. "Location utility-based map reduction". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2015, pp. 479–486. DOI: 10.1109/ICRA.2015.7139223.

[64] Cristian Tanas and Jordi Herrera-Joancomartí. "Crowdsensing Simulation Using ns-3". In: *Citizen in Sensor Networks: Second International Workshop, CitiSens 2013* (2014). Ed. by Jordi Nin and Daniel Villatoro. Springer International Publishing, pp. 47–58.

[65] CM Thomas and WE Featherstone. "Validation of Vincenty's formulas for the geodesic using a new fourth-order extension of Kivioja's formula". In: *Journal of Surveying engineering* 131.1 (2005), pp. 20–26.

[66] M. Usman, M. R. Asghar, I. S. Ansari, F. Granelli, and K. Qaraqe. "Towards Energy Efficient Multi-Hop D2D Networks Using WiFi Direct". In: *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*. 2017, pp. 1–7. DOI: 10.1109/GLOCOM.2017.8254045.

[67] Jing Wang, Jian Tang, Guoliang Xue, and Dejun Yang. "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.

[68] D. Willkomm, S. Machiraju, J. Bolot, and A. Wolisz. "Primary user behavior in cellular networks and implications for dynamic spectrum access". In: *IEEE Communications Magazine* 47.3 (2009), pp. 88–95. ISSN: 0163-6804. DOI: 10.1109/MCOM.2009.4804392.

[69] W. Wu, J. Wang, M. Li, K. Liu, F. Shan, and J. Luo. "Energy-Efficient Transmission With Data Sharing in Participatory Sensing Systems". In: *IEEE Journal on Selected Areas in Communications* 34.12 (2016), pp. 4048–4062. ISSN: 0733-8716. DOI: 10.1109/JSAC.2016.2621359.

[70] H. Xiong, D. Zhang, G. Chen, L. Wang, V. Gauthier, and L. E. Barnes. "iCrowd: Near-Optimal Task Allocation for Piggyback Crowdsensing". In: *IEEE Transactions on Mobile Computing* 15.8 (2016), pp. 2010–2022. ISSN: 1536-1233. DOI: 10.1109/TMC.2015.2483505.

[71] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi. "Internet of Things for Smart Cities". In: *IEEE Internet of Things Journal* 1.1 (2014), pp. 22–32. ISSN: 2327-4662. DOI: 10.1109/JIOT.2014.2306328.

[72] Haifeng Zhang and Yevgeniy Vorobeychik. "Submodular Optimization with Routing Constraints". In: *Conference on Artificial Intelligence (AAAI)*. 2016.

[73] Hongxu Zhang, Yufeng Wang, and Chiu C. Tan. "WD2: An Improved Wifi-direct Group Formation Protocol". In: *Proceedings of the 9th ACM MobiCom Workshop on Challenged Networks*. CHANTS '14. Maui, Hawaii, USA: ACM, 2014, pp. 55–60. ISBN: 978-1-4503-3071-8. DOI: 10.1145/2645672.2645674. URL: http://doi.acm.org/10.1145/2645672.2645674.

[74] Q. Zhao, Y. Zhu, H. Zhu, J. Cao, G. Xue, and B. Li. "Fair energy-efficient sensing task allocation in participatory sensing with smartphones". In: *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*. 2014, pp. 1366–1374. DOI: 10.1109/INFOCOM.2014.6848070.

[75] Z. Zheng, Y. Peng, F. Wu, S. Tang, and G. Chen. "Trading Data in the Crowd: Profit-Driven Data Acquisition for Mobile Crowdsensing". In: *IEEE Journal on Selected Areas in Communications* 35.2 (2017), pp. 486–501. ISSN: 0733-8716. DOI: 10.1109/JSAC.2017.2659258.

Online resources

[20] *CrowdSenSim*. Accessed march 5, 2018. 2016. URL: https://crowdsensim.gforge.uni.lu/index.html.

[35] *Internet of Things (IoT) Market: Global Demand Growth Analysis and Opportunity Outlook 2023*. Accessed February 3, 2018. 2017. URL: http://www.researchnester.com/reports/internet-of-things-iot-market-global-demand-growth-analysis-opportunity-outlook-2023/216.