TÉCNICO LISBOA

MultiTrack: Flexible Use of Multiple Location Technologies

Rui Diogo David dos Santos

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisors: Prof. Paulo Jorge Pires Ferreira
Prof. João Barreto

Examination Committee

Chairperson: Prof. Luís Manuel Antunes Veiga
Supervisor: Prof. Paulo Jorge Pires Ferreira
Members of the Committee: Prof. João Carlos Serrenho Dias Pereira

February 2017

Acknowledgments

I would like to thank my parents for their friendship, encouragement and caring over all these years, for always being there for me through thick and thin and without whom this project would not be possible. I would also like to acknowledge my dissertation supervisors Prof. Paulo Ferreira and Prof. João Barreto for their insight, support and sharing of knowledge that has made this Thesis possible.

Last but not least, to all my friends and colleagues that helped me grow as a person and were always there for me during the good and bad times in my life. Thank you.

Abstract

With the development of networks and mobile devices, Location-based Services (LBSs) have emerged that require location information to provide their services to users. Actually, there are several positioning systems capable of acquiring location information for these services. However, they all focus on particular scenarios by applying specific location technologies and/or techniques which they consider more suited for those cases. If we change the scenarios where these approaches are used, the accuracy of the information collected is highly affected, and so are the LBSs. Considering that LBSs are not constrained to any particular scenario and that they might have different requirements, we propose a different approach, called MultiTrack, which instead of focusing on specific scenarios and requirements, tries to accommodate them all by supporting the flexible usage of different location technologies and/or techniques by LBSs and by allowing new approaches to be easily added, thus contributing for LBSs to adjust the process of acquiring location information to best fit their requirements.

In this work, we start by describing the majority of the location technologies and/or techniques and some of the positioning systems that apply them, to understand their differences and determine which are the most valuable for providing location information for LBSs. Then, we present MultiTrack's architecture, enhancing how it supports the flexible usage of different location technologies and/or techniques by LBSs. For testing and evaluation purposes, we developed a smartphone application called Cycle-to-Shop, which requires location information for rewarding users that arrive at certain locations according to different rules and scenarios (e.g. go to school by bicycle). Experimental results obtained by using the Cycle-to-Shop rewarding application with MultiTrack are presented and described, enhancing how MultiTrack achieves its goal of supporting the flexible usage of different location technologies and/or techniques by LBSs and how that contributes for collecting the location information that best fit LBSs

requirements. At last, we present the conclusions obtained during the project.

Keywords

Location-based Services; positioning systems; location information; accuracy; energy; infrastructure; flexible; location technologies and/or techniques; MultiTrack; Cycle-to-Shop.

Resumo

Com o desenvolvimento das redes e dos dispositivos móveis, surgiram os serviços baseados em informação de localização, que necessitam de informação para prestar os seus serviços aos utilizadores. Actualmente, existem vários sistemas de posicionamento capazes de fornecer informação de localização a estes serviços. No entanto, estes tendem a focar-se em cenários específicos através da utilização de tecnologias de localização que melhor se adaptam aos mesmos. Se mudarmos os cenários onde estes sistemas são utilizados, a informação que eles adquirem é consideravelmente degradada, o que influencia a prestação dos serviços que dependem desta informação. Tendo em conta que estes serviços não estão restrictos a nenhum cenário ou requisito específico, não devemos focar-nos em cenários particulares mas sim, generalizar para abranger o maior número de situações possivel. É por este motivo que propomos o MultiTrack, um sistema de posicionamento que utiliza várias tecnologias de localização mas que permite a adição fácil e fléxivel de novas soluções, para que seja possível responder da melhor forma a novos cenários e requisitos, com o objectivo de fornecer aos serviços de localização, a informação que melhor se adequa às suas necessidades.

Neste trabalho começamos por descrever as tecnologias de localização existentes e os sistemas que as utilizam, para compreendermos quais as tecnologias que devem ser suportadas pelo MultiTrack. De seguida, apresentamos a sua arquitectura, realçando a forma como este permite a utilização fácil e fléxivel de várias tecnologias de localização e a integração de novas soluções. Para efeitos de testes, desenvolvemos a applicação Cycle-to-Shop, que é basicamente um serviço baseado em informacão de localização que premeia os utilizadores que chegam a determinadas localizações de acordo com diferentes regras, por exemplo, chegar à escola de bicicleta. Os resultados experimentais obtidos apartir da utilização do Cycle-to-Shop com o MultiTrack são descritos e analisados, realçando a forma como a utilização fléxivel das diversas técnicas de localização contribui para obter a informação de localização que melhor se adapta aos requisitos do Cycle-to-Shop e dos serviços baseados em informação de

localização em geral. Por fim, apresentamos as conclusões obtidas durante o trabalho.

Palavras Chave

informação de localização; sistemas de posicionamente; precisão; energia; infraestrutura; técnicas e tecnologias de localização; MultiTrack; Cycle-to-Shop.

Contents

List of Figures

List of Tables

Listings

Acronyms

AP	Access Point
BLE	Bluetooth Low Energy
DR	Dead Reckoning
GPS	Global Positioning System
GSM	Global System for Mobile Communications
IMU	Inertial Measurement Unit
IR	Infrared
LBS	Location-based Service
QR	Quick Response
MAC	Media Access Control
NFC	Near Field Communication
RF	Radio Frequency
RFID	Radio Frequency Identification
RSS	Received Signal Strength
TDOA	Time Difference of Arrival
TOA	Time of Arrival
WLAN	Wireless Local Area Network

1

Introduction

Contents

As wireless networks and mobile devices evolve, users demand more sophisticated applications. Location-based Services (LBS) arose as a major driver for delivering value to the wireless eco-system and device's location capabilities. These applications base their services on users' location information, which needs to be sufficiently accurate to allow them to provide their services correctly to users. Some of the most common LBSs include local news, navigation, points of interest, directory assistance, fleet management, asset tracking, location-sensitive building and local advertisement.

For these applications to provide their services, first they need to collect users' location information. Then, the service itself is provided either offline, i.e. based only on local information, or online, i.e. also based on remote information. An example of a service that can be provided offline is navigation, if maps used to navigate users are stored locally. However, if the maps are stored remotely, the service becomes online. Nowadays, most LBSs provide their services with an online component in order to take advantage of higher-level information, i.e. structured validated information. A typical online system is comprised by two stages: i) collecting and sending the location information to a server, and ii) querying the server to obtain higher-level information.

Both offline and online LBSs require sufficiently accurate location information, so that they can correctly provide their services to users. However, increasing the accuracy of this information has costs, usually in terms of energy and infrastructure. Energy costs are related with the energy spent by the devices that run these services and, since those devices are mobile and suffer from battery constraints, energy costs need to be minimal. Infrastructure costs, i.e. costs associated with acquiring, deploying and maintaining the required equipment for LBSs to provide their services to users, need also to be minimal, to ease their development and increase profits.

Having this trade-off into account, we cannot simply deliver location information with maximum accuracy for LBSs. Instead, we should adjust the accuracy of the information to best fit LBSs requirements, thus allowing them to provide their services correctly to users but with minimal costs (energy and infrastructure).

There are several positioning systems [1–8] that are able to provide location information for LBSs with different characteristics in terms of accuracy and costs (energy and infrastructure). These systems focus on specific scenarios, e.g. indoors or outdoors, by applying specific location technologies and/or techniques[1] to obtain location information with the highest accuracy and lowest energy and infrastructure costs. However, if we use them at different scenarios, their results are highly affected, i.e. either the accuracy of the collected location information degrades or the infrastructure costs grow. For example, the Global Positioning System (GPS)[2] focus essentially outdoors, mainly due to the usage of Radio Frequency (RF) waves and satellites strategically placed all around Planet Earth, allowing GPS receivers

[1] At a high level, location technologies and/or techniques are processes to determine users' location information. In Chapter 2 we describe these processes in more detail.

[2] http://www.gps.gov/

to acquire location information with reasonable accuracy (less than 10 m) and low infrastructure costs theoretically all over Earth's surface. However, for indoors, GPS is not a viable approach because its accuracy is highly degraded by physical structures, which are frequent in those scenarios.

Positioning systems based on Infrared (IR), Ultrasound, Radio Frequency Identification (RFID), Bluetooth[3] and Quick Response (QR) codes are examples of schemes that are usually more accurate than GPS but require specialized infrastructure, and due to that, are used mainly indoors (e.g. a small house), since the infrastructure costs associated with deploying such systems outdoors (e.g. a city) are huge.

The proliferation of static RF transmitters such as Wi-Fi APs and Global System for Mobile Communications (GSM) towers has enabled localization without the need for additional infrastructure, since these infrastructures are already widely deployed for other services, like Internet, IP Telephony and so on. The basic approach is to fingerprint each location in the space of interest with a vector of Received Signal Strength (RSS) measurements of the various transmitters. A mobile device is then localized by matching the observed RSS reading against the stored measurements. There are many positioning systems [1, 4] that apply this technique and are able to collect accurate location information (less than 3 m). However they all entail a considerable amount of manual effort to perform detailed measurements. There are other approaches, like EZ [3], SAIL [6] and Place Lab [5], which focused on decreasing these efforts while maintaining the accuracy of the previous approaches. Nevertheless, these works are based on Wi-Fi networks, and due to that, are used mainly indoors where these networks are widely available.

As we can see, there are many different approaches when it comes to localization and all can be potential solutions for providing location information for LBSs. Having this into account and considering that LBSs requirements may vary and are not known in advance, we cannot simply choose one of these approaches to deliver the location information that LBSs require, i.e. that best fit their requirements. Instead, we should support the flexible usage of different approaches and allow LBSs to control them, contributing for LBSs to adjust the process of collecting location information to best fit their requirements. This way, a higher number of LBSs would be able to correctly provide their services to users with minimal energy and infrastructure costs.

There are already several native location services provided by mobile operating systems[4], that allow LBSs to control different location technologies and/or techniques, usually through APIs, and adjust the process of collecting location information to best fit their requirements. The Fused Location Provider API[5], is an example of such APIs, since it allows Android LBSs to use and manage GPS, Wi-Fi and mobile sensors (accelerometer, gyroscope and magnetometer). However, there are several location technologies and/or techniques, which are not supported in those APIs and that might be useful for LBSs to collect the location information that best fit their requirements, thus allowing them to correctly

[3] https://www.bluetooth.com
[4] Android, iOS
[5] https://developers.google.com/android/reference/com/google/android/gms/location/FusedLocationProviderApi

provide their services to users with minimal energy and infrastructure costs, e.g. Bluetooth, QR-codes and Near Field Communication (NFC). If we imagine a LBS that needs to detect users' position with minimal costs, on an environment where wireless communications are unavailable, the usage of QR-codes might be a viable solution. As we can see, supporting the flexible usage of a wider range of location technologies and/or techniques allows a wider range of LBSs to collect the location information that best fit their requirements. This is where our work is focused in.

The goal of this work is then to support the flexible usage of different location technologies and/or techniques by mobile smartphone LBSs, to allow them to apply the most accurate and less energy and infrastructure demanding solution for each users' scenario, thus allowing them to provide their services to users with minimal energy and infrastructure costs.

To achieve it, we face essentially one challenge: the diversity of LBSs requirements and environments, where they require location information. For example, there are LBSs that require location information, both indoors and outdoors, with an accuracy at the meter level, while others require location information only indoors, but with an accuracy at the centimeter level.

Our solution is called MultiTrack. MultiTrack is a framework that uses different location technologies and/or techniques to collect location information. Moreover, it allows LBSs to control at any moment, the location technologies and/or techniques used to collect the location information. When LBSs interact with the framework, they choose the location technologies and/or techniques as well as other important settings. An example of a setting that can be managed by applications is the rate at which the location information is acquired, which influences the amount of information collected and consequently, the energy consumed by the device. Figure 1.1 illustrates a LBS managing the framework to collect users' location information using GPS when the user is outdoors, and then Wi-Fi and mobile sensors when the user goes indoors.

To achieve the goal of supporting the flexible usage of different location technologies and/or techniques by mobile smartphone LBSs, MultiTrack has the following requirements:

- **Modularity**, i.e. Location modules that implement the location technologies and/or techniques supported need to be easily added and removed.

- **Technologies Control**, i.e. LBSs require the ability to control the supported location technologies and/or techniques, to acquire the location information that best fit their requirements.

- **Energy Efficiency**, i.e. the energy spent by the framework to collect location information need to be minimal since it runs on mobile devices, which are battery constrained.

To test if MultiTrack achieves its goal of supporting the flexible usage of different location technologies and/or techniques by mobile smartphone LBSs, we developed a LBS called Cycle-to-Shop.

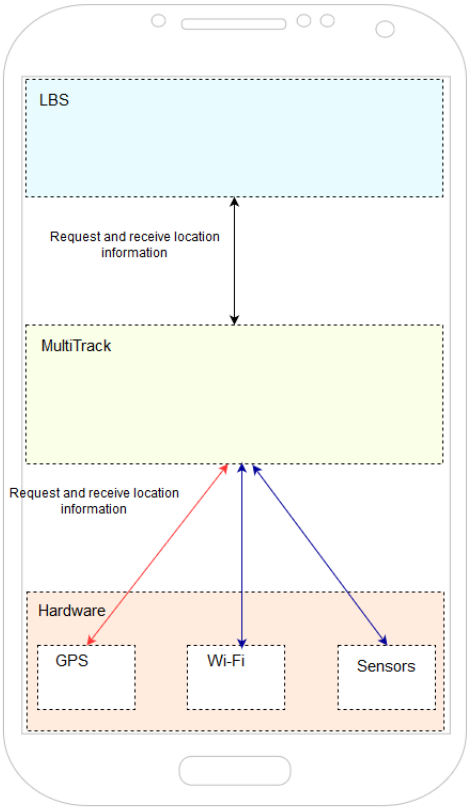

Figure 1.1: Example of an application managing MultiTrack to collect location information with different location technologies and/or techniques, according to users' current environment.

1.1 Cycle-to-Shop

The Cycle-to-Shop rewarding application is an online LBSs that runs in smartphones and is associated with the TRACE project[6], which aims at assessing the potential of movement tracking services to better plan and promote walking in cities, and developing tracking tools to fuel the take up of walking and cycling measures.

Cycle-to-Shop requires location information to motivate its users to cycle more frequently. It rewards them based on the distance they do and on the places they have been, which are usually stores from partners that support the application. Cycle-to-Shop relies on 3 basic assumptions: i) users carry mobile smartphones with location capabilities, e.g GPSs, Wi-Fi and Bluetooth, ii) users can be either indoors or outdoors, and iii) every store has a Bluetooth beacon inside.

Moreover, it requires location information with an accuracy ideally better than 10 m, both indoors and outdoors, to correctly determine the distance that users do while cycling (outdoors), and to locate them

[6] http://h2020-trace.eu/

at the right stores (indoors). To make sure that users are rewarded when they are at the right stores, each store has a Bluetooth beacon inside that can be used to locate users nearby, i.e. inside the store. The addition of these beacons is crucial because GPS and Wi-Fi might not be always available at the stores, e.g. when Wi-Fi is not available and GPS is blocked by the walls or obstacles in general.

Another important requirement is the detection of users' modality, since Cycle-to-Shop only rewards users if they are cycling. At last, it requires minimal energy consumption, since it runs on mobile smartphones, which are battery constrained, and also minimal infrastructure costs, to ease its development and acceptance.

In the next Chapter, we discuss the related work, describing in a broad manner the main technologies and/or techniques used in localization and, in more detail, the positioning systems that implement them to collect location information. The idea is to understand their differences to determine which should be supported in MultiTrack.

In Chapter 3 we describe the MultiTrack architecture, focusing on its main components and interactions, enhancing how they support the flexible usage of different location technologies and/or techniques by LBSs, and ease the addition of new ones.

In Chapter 4 we describe the options available and the choices made for implementing the location technologies and/or techniques chosen on Chapter 2, as well as the core components of MultiTrack, enhancing their advantages and disadvantages at the light of MultiTrack's requirements.

In Chapter 5 we describe the experiments done using the Cycle-to-Shop rewarding application with MultiTrack to collect location information. The objective of these experiments is to determine if MultiTrack achieves the goal of supporting the flexible usage of different location technologies and/or techniques by mobile smartphone LBSs. At last, we present the experimental results obtained and corresponding conclusions.

Finally on Chapter 6, we present th conclusions and MultiTrack's limitations and future work.

2

Related Work

Contents

This work focuses on developing a framework that supports the flexible usage of different location techniques and/or technologies by mobile smartphone LBSs, to allow them to apply the most accurate and less energy and infrastructure demanding solution for each users' scenario. In this Chapter, we describe in a broad manner, the main technologies and/or techniques used in localization and, in more detail, the positioning systems that implement them to collect location information. The idea is to understand their differences to determine which should be supported in MultiTrack.

2.1 Localization Techniques

Localization techniques can be divided into two big families: i) network side localization, where the network is itself responsible for tracking the individual devices and; ii) device-centric localization, where the device probes its environment to determine its own position. Since network side localization requires the deployment of specialized infrastructure [9], which are not always available, we will focus on the device-centric localization techniques that only require a smartphone to be applied. The device-centric localization techniques can be further categorized as follows: i) Fingerprint based, ii) Received Signal Strength based, iii) Proximity based, iv) Time based and, v) Dead Reckoning. Next, we briefly explain each of these techniques and some positioning systems that actually use them.

2.1.1 Fingerprint based

The fingerprint based approach is equivalent to scene analysis or pattern matching [10] and usually it consists on two phases: i) the offline phase, and ii) the online phase.

In the offline phase, the location fingerprints are collected by performing a site-survey of the received RSS from multiple APs. Basically, there are 2 different types of techniques to collect these fingerprints: i) deterministic techniques and, ii) the probabilistic techniques. The deterministic techniques represent the signal strength of an AP at a location by a scalar value, for example, the mean value, and use non-probabilistic approaches to estimate the user location. Examples of systems using these techniques are Radar, where nearest neighborhood techniques are used to infer users position. On the other hand, probabilistic techniques store information about the signal strength distributions from the APs in the radio map and use probabilistic approaches to estimate users location. For example, Nibble system uses a Bayesian Network approach to estimate user location.

When the location fingerprints are collected, the entire area is covered by a rectangular grid of points, and for each point, the corresponding RSS values from APs are determined and stored in a database or in a table. The vector of RSS values at a point on the grid is called the location fingerprint of that point.

During the online phase, the client trying to determine its position takes a sample of the signal strengths of the APs in range at his current location and sends it to the server. Then, the server feeds

the sample vector collected to an algorithm that compares it against the fingerprints stored and outputs the closest match as the users' coordinates.

The most common algorithm to estimate the location computes the Euclidean distance between the measured RSS vector and each fingerprint in the database. In this method, the algorithm takes the sample vector and simply calculates the distance between each well-known fingerprint and the sample, and outputs the location of the fingerprint with the smallest distance (see equation (2.1)). It usually runs on the server where the fingerprints are located, to leverage computational power.

$$d = \sqrt{(s_1 - f_1)^2 + (s_2 - f_2)^2 + \ldots + (s_n - f_n)^2} \tag{2.1}$$

where: d is the distance (or error) between the sample and the fingerprint, $s = \{s_1, \ldots, s_n\}$ is the sample, and $f = \{f_1, \ldots, f_n\}$ is the fingerprint.

Examples of positioning systems based on this localization technique are RADAR [1] and Horus [4].

2.1.2 Received Signal Strength based

The RSS based approach uses the radio propagation behavior of the signal to determine the distance between receivers and emitters. If we consider two points, A and B, and that in point A the RSS is higher than in point B, it means that point A is closer to the emitter than point B. This way, we can estimate the distance between receivers and emitters.

2D trilateration is often used to compute the location of the receiver using the RSS and position of 3 or more emitters. With one emitter we can define that the possible receiver's location lie on a circle with the emitter as its center. The second emitter allows the position of the receiver to narrow down to the intersection of both circles. A third emitter gives a two-point receiver's position, which is the intersection of the three circles, each one centered on its corresponding emitter, as shown on Figure 2.1.

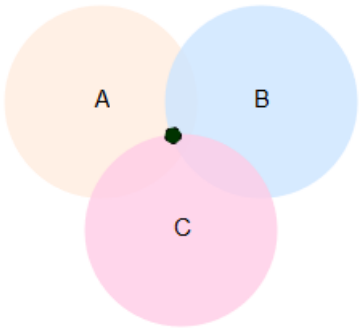

Figure 2.1: 2D trilateration using emitters A, B and C to obtain receivers position.

Examples of positioning systems based on Signal Strength technique are EZ [3] and [11].

2.1.3 Proximity based

The idea of proximity or ad-hoc based approaches is that if a node is within the range of a beacon (e.g., Bluetooth) or tag (e.g., a QR code), then it is coarsely located at the beacon's position. Obviously, this position is accompanied by an error, which depends on the technology used, but is generally very low and hence the advantage of using this method of location.

Examples of positioning systems based on this method are Labelee [12], QR-Maps [8], Active Badge [13], Place Lab [5], Foursquare[1] and RFIDLocator [14].

2.1.4 Time based

Time based approaches use a measurement of time between sending and receiving of signals to estimate the distance between emitters and receivers. Two of the most relevant techniques are: i) Time of Arrival (TOA) or **TOF!** (**TOF!**) with trilateration, and ii) Time Difference of Arrival (TDOA) with multilateration.

TOA uses the absolute time of arrival at a certain receiver. When an emitter broadcasts a signal at t_1 and a receiver gets it at t_2, the receiver uses the difference between t_2 and t_1 to calculate the distance between him and the emitter, using the following formula:

$$d = c \times (t_2 - t_1) \tag{2.2}$$

where,
c is the speed of light
t_1 is the true time of the signal transmission from the emitter
t_2 is the true time of the signal arrival at the receiver.

However, this method requires clock synchronization between emitters and receivers, and since clocks are never perfectly synchronized, there are always errors in the measurements that degrade the accuracy of receiver's position.

Examples of positioning systems based on TOA are TELIAMADE [15], Active BAT[2], Cricket[3] [16] and GPS [2].

TDOA does not require clock synchronization between emitters and receivers, only emitters have to be synchronized. Receivers measure the difference of times of signals emitted from three or more synchronized transmitters at known locations. The difference of times yields a difference of distances between those stations, allowing receivers to estimate their own positions using multilateration.

[1] https://foursquare.com/
[2] http://www.cl.cam.ac.uk/research/dtg/attarchive/bat
[3] http://cricket.csail.mit.edu

2.1.5 Dead Reckoning

Dead Reckoning (DR) or Ded Reckoning for deduced reckoning is a relative navigation technique that calculates one's current position by using a previously determined position and adding up successive position displacements, that can be in the form of changes in Cartesian coordinates(i.e. x and y coordinates) or, more typically, in heading and speed or distance. These position displacements are calculated by using mobile devices as Inertial Measurement Units (IMUs), using their accelerometers, gyroscopes and the magnetometers. With sufficiently frequent absolute position updates, the linearly growing position errors of DR can be contained within pre-defined bounds.

Examples of positioning systems based on DR technique are Pedestrian Dead Reckoning [7], SAIL [6] and [].

2.2 Positioning Systems

There are several positioning systems that apply the previous location techniques to collect location information. In this section, we describe these systems, enhancing their characteristics and experimental results.

2.2.1 RADAR

RADAR is the world's first Wi-Fi [17] signal-strength based indoor positioning system. It is a device-oriented RF based system that uses the Fingerprint based location technique for locating users indoors. To acquire users' position, their wireless devices measure the RSS from APs within their range and then search the radio map to determine the signal strength entry that best matches the measured signal strength. To improve it's estimate, RADAR takes into account the recent movement history of the user and dynamic changes such as temperature, the number of people present, and any other environmental factors which will effect the radio map. Experimental results have shown that despite the hostile nature of the radio channels, it is possible to locate and track users with a high degree of accuracy. The median resolution of the RADAR system is in the range of 3 m in a typical office room.

RADAR is a viable option to collect users' location information indoors because it's where Wi-Fi networks are more developed. Taking advantage of such a wide network, which was already deployed for other services like Internet, multimedia applications and IP telephony, decreases system's infrastructure costs, despite the high calibration costs from building detailed RF maps. Moreover, Wi-Fi networks are evolving and their APs are increasing in number and getting closer to users' devices, thus improving RADAR's accuracy.

For outdoors, these networks are also growing and becoming more available, but they have limita-

tions, because users often need to pay to use them. Moreover, using Wi-Fi in mobile contexts is a power consuming solution, since users are constantly moving and connecting to different APs, thus increasing device's power consumption [18].

2.2.2 Horus

Similarly to RADAR, Horus is a Wireless Local Area Network (WLAN) location determination system that uses the RF signals from APs to determine users' position using the fingerprint probabilistic location technique. The design of the Horus system aims at satisfying two goals: i) high accuracy, and ii) low computational requirements. To achieve these goals, Horus identified several causes for the wireless channel variations and implemented different techniques to solve them, thus improving the obtained results in terms of accuracy when comparing to previous works, e.g RADAR. Moreover, it uses location-clustering techniques to reduce the computational requirements of the algorithm, thus allowing it to be implemented in a greater variety of energy constrained devices.

Horus has an error of less than $0.6\,\mathrm{m}$ on the average and its computational requirements are more than an order of magnitude better than other WLAN location determination systems. Moreover, the techniques developed in the context of the Horus system are general and can be applied to other WLAN location determination systems to enhance their accuracy. For example, several Horus techniques were applied to RADAR and the average distance error was reduced by approximately 50%.

Despite the obtained results, Horus suffers from the same limitations as RADAR. An arising issue with Horus system is that it is a calibration intensive solution, which increases the overall infrastructure costs. These costs are related with building the RF maps, required to apply the fingerprint location technique. One possible way to mitigate this problem is to automate the radio-map generation process, which is an actual research area.

2.2.3 EZ

EZ is an indoor positioning system that leverages existing infrastructure without requiring any explicit pre-deployment effort. It is based on 3 basic assumptions: i) there are enough Wi-Fi APs to provide excellent coverage throughout the indoor environment, ii) users carry mobile devices, such as smartphones or tablets, equipped with Wi-Fi, and iii) occasionally a mobile device obtains an absolute location fix, say by obtaining a GPS lock at the edges of the indoor environment, such as at the entrance or near a window.

To locate users indoors, EZ applies the RSS location technique using users' mobile devices to record the signal strength from at least 3 Wi-Fi APs in range, along with the occasional location fix from GPS when available, and send this data to a central localization server. The server uses it to simultaneously

learn the characteristics of the RF propagation environment and to determine the distance between mobile devices and Wi-Fi APs. These distances allows for only one possible realization for their relative positions, which is performed in terms of absolute latitude and longitude coordinates.

EZ is an indoor positioning system that locates users using mostly the Wi-Fi infrastructure nearby, which is usually available indoors without restrictions but rarely outdoors. The key advantages of EZ over other works [4] is that it does not require any pre-deployment effort or user participation in the localization process, thus reducing calibration costs. Moreover, EZ yields a median localization error between 2 m and 7 m, varying with the indoors area, which is only somewhat worse than the 0.7 m and 4 m yielded by the best-performing but calibration intensive Horus.

2.2.4 Labelee

Labelee is an indoor positioning and orientation system, which optimizes the user mobility inside buildings. Labelee uses QR codes and NFC chips that act as information points to the users, which only have to use their smartphones to take a picture of one of those labels or read the chips to be placed into a map, showing their current position.

QR codes are two dimensional barcodes consisting of black modules arranged in a square pattern on a white background. They were designed for fast readability and high capacity of information (up to 7089 characters). Moreover, they have the ability to error correction, which allows data to be decoded even if the code is partly dirty or destroyed.

NFC was specially designed to be used by devices within close proximity to each other (10 cm or less). Devices using NFC may be active or passive. A passive device is called tag and contains information that other devices can read but does not read any information itself. Active devices can read and write information, e.g, smartphones. To ensure security, NFC often establishes a secure channel and uses encryption when sending sensitive information. It is typically used for mobile payments and for promotional purposes. Examples of positioning systems that use this technology are Labelee [12], Foursquare[4] and NFC Internal [**?**]. In all these systems, users are located by approaching their smartphones to the tags, read the data from them and send it to a location server. When the server receives the message containing data from a tag, it identifies the tag where the user have read the data and determines his position by proximity to the tag.

2.2.5 QR-Maps

QR-Maps is an indoor positioning system that uses QR codes and Google Maps API[5] to locate users, which only have to take a picture of one those codes, using their smartphones, to decode it's information

[4] https://foursquare.com/
[5] http://code.google.com/api/maps/index.html

and send it to a location server using an Internet connection. When the server receives this information, it returns an URL that shows a map centered at the specified code location.

2.2.6 Active Badge

The Active Badge system is one of the first IR based positioning systems. IR signals are invisible electromagnetic radiation with longer wavelengths than those of visible light, meaning that they are invisible to human eye. However, IR waves can be detected with specific instruments like nigh-vision goggles or IR cameras. A typical use for IR energy is the television remote control. Humans cannot see the energy released by the remote control but the television detects it and reacts accordingly. However, these signals require line-of-sight between transmitters and receivers, otherwise they are interrupted and do not reach their destination.

The Active Badge positioning system was designed at AT&T Cambridge in 1990s, to cover the area inside a building and provide symbolic location information of each badge such as the room where it is. The Active Badge system uses commonly available IR technology to locate the badges. The system assumes that every person has a badge and by locating that badge, the system locates the person in its coverage area. Badges transmit a globally unique IR signal every 15 seconds. In each room, one or more sensors are fixed to detect the signal. The position of the badge can be specified by the information from these sensors, which are connected with wires and forward the information to a central server, which is then queried to obtain badge's position. Badges were designed in a small package roughly $55x55x7\mathrm{mm}$ and weighs a comfortable $40\,\mathrm{gm}$. Although the price of these badges and networked sensors are cheap, the cables connecting sensors raise the cost of the Active Badge system.

Labelee and QR-Maps focus on collecting users' location information indoors, when wireless infrastructures are nonexistent or wireless communications suffer high interferences, due to infrastructure costs associated with acquiring the required codes and chips and deploying them. Nevertheless, they can provide accurate location information, using the Proximity location technique, with low energy costs. For outdoors, these systems are not viable approaches due to high infrastructure costs.

Active Badge focus on optimizing accuracy rather than wide-scale deployment. It offers an accuracy of $7\,\mathrm{cm}$, but it requires a costly infrastructure, due to infrared characteristics, e.g line-of-sight between emitters and receivers and low range. Moreover, infrared infrastructure is not already widely deployed, as other wireless infrastructures, e.g Wi-Fi, meaning that it needs to be built from start, thus increasing infrastructure costs.

2.2.7 Place Lab

Place Lab is a positioning system that uses RF beacons, such as 802.11APs, with GSM cell phone towers and Bluetooth devices already existent in the environment, to locate users by applying the prox-

imity location technique. Users carry mobile devices with them, such as notebooks and smartphones that receive and interpret beacon's signals. Since each signal has an unique ID, they compute their own location by hearing one or more beacons, looking up the associated beacons' positions in a locally cached map, which is built from surrounding databases, and estimating their own position referenced to the beacons' positions. Place Lab's main objectives are: i) maximizing coverage, which is measured by the percent of time location fixes are available in people's daily lives, and ii) providing a low barrier to entry for users and developers.

System's coverage is directly related with users location and since they consider that users spend most of their time indoors, they also tend to focus indoors. That's why Place Lab bases it's localization scheme on indoor widely deployed infrastructures, e.g Wi-Fi. Moreover, the databases that hold beacon's location information come from institutions that own a large number of beacons, e.g companies and universities that often know their locations since that information is commonly recorded as part of a deployment and maintenance strategy. As we have seen in previous works, using infrastructures that are already deployed for other ends helps reducing system's infrastructure costs and increase it's availability.

To provide a low barrier entry to users, Place Lab uniquely identifies each beacon's signal and stores beacon's location information in databases that are widely spread and easily reached by users. This way, they do not need to transmit data to determine their location nor listen to other users' data transmissions. In the case of 802.11, receiving beacon's signals can be done entirely passive by listening for the beacon frames periodically. These beacon frames are sent in the clear, and are not affected by Media Access Control (MAC) address authentication. In Bluetooth case, it requires clients to initiate a scan in order to find nearby beacons. However, due to restricted programming interfaces, detecting GSM cell IDs requires handsets to associate with nearby cell towers, thus increasing the barrier entry for users. Moreover, the accuracy provided using this system is around 20 and 30 m, which may not enough for many LBSs. Nevertheless, Place Lab offers high availability and low infrastructure costs.

2.2.8 RFIDLocator

RFIDLocator is a RF based positioning system that uses RFID[6] tags to locate users or any objects that carry them. RFIDLocator is basically composed by 3 parts: i) the readers, ii) the tags, and iii) the monitoring information system. Readers continuously broadcast RF signals, within a certain range, that are received and interpreted by tags. Tags receive these signals and reflect them to readers, adding information to the signal that allow readers to identify them. When readers receive these responses, they automatically inform the monitoring information system about tags location, which is determined by applying the proximity location technique.

[6] http://www.rfidjournal.com/

Positioning systems based on RFID like RFIDLocator require high infrastructure costs because the infrastructure needs to be deployed from start, since RFID infrastructures are not widely spread and available. Moreover, the accuracy of the system is determined by the number of readers in the area and the range of those readers plus environmental factors that can also influence the RF signals. Having this into account, the use of such systems are limited to indoors, e.g locate users inside an university or company's building because for outdoors the costs are huge. Nevertheless, they provide accurate location information, since tag's location is determined by proximity to the readers. Moreover, the energy costs related with the process of acquiring this information are low, when compared to other solutions like Wi-Fi or GPS.

2.2.9 ZONITH

ZONITH is an indoor Bluetooth-based positioning system that locates any Bluetooth devices, e.g smartphones, registered on the system with an accuracy of 2 m.

Bluetooth[7] is a wireless communication technology that connects different types of devices, such as smartphones, tablets and peripherals with computers. Bluetooth was designed for mobile constrained devices, as a secure, short-range communications technology, and nowadays, most of them have Bluetooth modules incorporated that allows them to communicate with each other with a range of 10 m. For these reasons, it has been a popular choice for communications between mobile devices.

The device positioning is achieved by using a network of strategically mounted beacons that use Bluetooth Low Energy (BLE) to detect bluetooth devices nearby and monitor them by applying the RSS location technique.

BLE is a newer version of Bluetooth that was designed to have a very fast connection set-up (a few milliseconds) and exchange short messages, while maintaining the security and robustness properties of the regular Bluetooth. BLE was designed to be used with the Internet of Things, where transmitting short bursts of data is perfect for most smart devices. It cannot support audio streaming, but in most of these scenarios, it is not actually needed. BLE is part of Bluetooth, so dual-mode devices i.e devices that operate both in BLE and Bluetooth mode, such as smartphones, are able to communicate with Bluetooth and BLE devices at the same time.

The main advantage of using BLE beacons over regular Bluetooth beacons, is the reduction of device's power consumption, which is a major advantage since most devices, e.g smartphones, are power constrained. This reduction in power consumption is possible because BLE is optimized for short burst of data, which are usual in this type of scenarios, and it does not require both devices to be paired, as Bluetooth does. This increases device's power savings but at the cost of security. Fortunately, for many applications, it makes sense loosing security to improve power consumption, since the exchanged data

[7] https://www.bluetooth.com

might not be critical. Nevertheless, the usage of these beacons comes with high infrastructure costs, which grow with the area covered since these beacons need to be acquired, deployed and maintained. For this particular reason, BLE beacons are usually applied for indoor scenarios, where the covered areas are smaller and costs lower.

2.2.10 LANDMARC

Similarly to RFIDLocator, LANDMARC is a RF based positioning system that uses RFID technologies to locate users indoors. However, LANDMARC used a different approach in order to increase the accuracy without deploying more readers in the environment. For this, it applies the RSS location technique by using extra location reference tags to help in location calibration. These reference tags serve as reference points in the system, the so-called ”landmarks”. This approach has 3 major advantages: i) replaces expensive readers by these active tags, which are cheaper, thus decreasing system's infrastructure costs, ii) the environmental dynamics can easily be accommodated, thus overcoming many environmental factors that contribute to variations in detected range because the ”landmarks” are subject to the same environmental effects as the tags to be located, and iii) the location information is more accurate and reliable.

However, there are several problems related with RFID technology. First, none of the currently available RFID products provides the signal strength of the tags directly. Instead, the reader reports ”detectable” or ”not detectable”. This problem comes from the fact that RFID is a location technology that was not initially designed for indoor location sensing. This forces LANDMARC to spend approximately 1 minute to estimate the signal's strength, which also increases errors in the measurements.

The second problem is the long latency between a tracking tag being physically placed to its location being computed by the location server. There are 2 reasons for this problem: i) the time required to estimate signal's strength, as previously described, and ii) the time interval of emitting 2 consecutive IDs from an active tag, which is 7.5 s to avoid collisions.

The third problem is the variation of tag's behavior. IN LANDMARC's approach, the basic assumption is that all tags have roughly the same signal strength in emitting the RF signal, but experimental results show that the power level detected by the same reader from different tags in an identical location may be different. A possible explanation for these differences is the variation of tag's chips, circuits and batteries.

Experimental results show that using 4 RF readers and one reference tag per square meter, LANDMARC can determine users' location with accuracy between 1 and 2 m.

LANDMARC proves that RFID is viable option for indoor localization because it has several advantages when comparing with other technologies, like Infrared or QR-codes, e.g does not require contact or line-of-sight between emitters and receivers. However, it requires high infrastructures costs.

2.2.11 TELIAMADE

TELIAMADE [15] is a positioning system based on TOA of ultrasonic signal to estimate the distance between a receiver node and a transmitter node. TELIAMADE system consists of a set of wireless nodes equipped with a radio module for communication and a module for the transmission and reception of ultrasound signals. These signals are transmitted using a carrier frequency of 40 kHz and the TOF measurement is estimated by applying a quadrature detector to the signal obtained at the A/D converter output. The distance is calculated from the TOF taking into account the speed of sound. An excellent accuracy in the estimation of the TOF is achieved using parabolic interpolation and signal phase information. Experimental results show a location accuracy of 10 cm.

2.2.12 Active BAT

Active BAT is a positioning system based on TOA of ultrasonic signals. It was designed by researchers at AT&T Cambridge that provides 3-D position and orientation information for the tracked tags and which is meant for accurate indoor localization. It proposes the use of ceiling mounted beacons, connected to a central station, placed in order to maximize likelihood of line of sight to beacons. The system works by equipping target objects (bats) with a badge that emits ultrasound pulses. First, bats notify beacons about their presence, which is also known as registration. Then, beacons might send queries to the registered bats, which sends ultrasound pulses in response. If these pulses are detected by three or more beacons, the position of the bat is calculated on the central station, using multilateration. However, the performance of this system is highly influenced by the obstacles between beacons, central station and bats. Moreover, deploying a large number of sensors on the ceiling in each room is a time-consuming task, which degrades the scalability of this system. One of the risen problems in adopting Active BAT is related to a lack of scalability, since multiple beacons need to be well separated and correctly placed to be queried concurrently without the risk of interference, which results in complex and costly installation.

2.2.13 Cricket

Cricket is an indoor positioning system for pervasive and sensor-based computing environments developed at the Computer Science and Artificial Intelligence Laboratory of Massachusetts Institute of Technology, which is able to provide fine-grained location information (space identifiers, position coordinates, orientation) to applications running on laptops, sensor nodes or mobile devices, using both Ultrasound and, occasionally, RF signals. The Cricket system is composed by two components: i) ultrasound emitters, and ii) ultrasound receivers. Ultrasound emitters are placed on walls or ceilings at known positions and the receivers are the devices that are tracked. Like TELIAMADE and Active BAT, Cricket

system uses the TOA measuring method and triangulation location techniques to locate the receivers. A peculiar feature of this approach is the fact that it provides some privacy to the user, since it performs all the position triangulation calculation locally, in the receiver. This implies that the receiver holds its location information and can decide how and where to publish it. Moreover, emitters also transmit RF messages for time synchronization of the TOA measurements, which contain their location information. Thus when not enough emitters are available for the triangulation location calculation, the receiver can use the additional RF messages to get proximity location information. The Cricket system addresses the issues of fault tolerance by using radio frequency signals as a second method of proximity positioning in the case of not enough emitters being available. Unlike the Active BAT system, which uses a grid of receivers connected to a central station, the Cricket system uses a reduced number of emitters fixed on the ceiling, because the target object receives and processes the ultrasound signals to locate itself.

TELIAMADE, Active BAT and Cricket are ultrasound positioning systems based on the TOA of ultrasonic signals to estimate the distance between targets and infrastructural nodes.

Ultrasound is an oscillating sound pressure wave with a frequency that is greater than the upper limit of the human hearing range, varying from 20 khz up to several gigahertz. Ultrasound signal is characterized by a slow propagation speed, a negligible penetration in walls and a low cost of the transducers. Moreover, it is capable of reaching an accuracy of few centimeters.

TELIAMADE, Active BAT and Cricket focus on optimizing accuracy rather than wide-scale deployment. They can offer accuracies between 5 and 50 cm but, since they all require hardware infrastructure to be installed in the environment, they are generally expensive in terms of infrastructure costs, costing thousands to tens of thousands of **US!** (**US!**) dollars for a 1000 m^2 installation.

Having this into account, we can conclude that these systems are better options to locate users indoors when high accuracy is required. For outdoors or in situations where a lower accuracy is not an issue, these systems are not viable when compared other approaches, e.g GPS and EZ, mainly due to their high infrastructure costs. This happens due to Ultrasound slow propagation speed, negligible penetration in walls and the fact that Ultrasound infrastructure is not widely deployed as other infrastructures, e.g Wi-Fi. Moreover, since they are based on TOA to estimate the distance between targets and infrastructural nodes, their clocks need to be synchronized.

2.2.14 Global Positioning System

GPS is a satellite based navigation system developed by the **US!** Defense Department. It consists of 27 satellites (as of May 2005) that orbit the earth. These satellites follow well-known orbits, so their positions are predictable, and each satellite transmits a RF signal encoded with a unique bit pattern, that allows GPS receivers to identify them. The idea behind GPS is to calculate the distance between the receivers and the satellites, based on TOA, to acquire the location information, which is composed

by current position and velocity of the receiver plus the time instant at which they were collected. For GPS receivers to acquire this information, they must calculate the distance between them and at least 4 different satellites, which is possible to do theoretically in any place on Earth because receivers are supposed to detect at least 4 different satellites at anytime in anyplace. GPS works world-wide and it provides a median accuracy of 10 m, which can be improved with various augmentation schemes.

For outdoors, GPS is the standard positioning system due to its high availability, accuracy and low infrastructural costs, despite the high energy consumption associated, which is currently the major problem of this system to be used by mobile devices. For indoors and in ”urban canyons” formed by tall buildings, GPS offers poor availability and degraded accuracy, since GPS receivers require a clear view to at least 4 satellites, which is hard to obtain.

2.2.15 SAIL

SAIL is an indoor positioning system that uses Wi-Fi APs together with smartphone sensors to estimate users' position. First, SAIL computes the distance between users' smartphones and an AP, using the propagation delay of the signal traversing between both by applying the TOA location technique. Second, it combines that distance with smartphone dead-reckoning location techniques, which basically consist on using the accelerometer to compute users' walking distance and the gyroscope to determine orientation changes. Third, it employs geometric methods to yield users' location using a single AP. Evaluations have shown that SAIL can capture users' location information with a mean error of 2.3 m.

SAIL focus on decreasing infrastructure costs for Wi-Fi based positioning systems, while maintaining a good accuracy in the information collected. Previous works, e.g EZ, do not require any calibration intensive tasks, e.g building detailed RF maps, but they require a high density of APs in the area, thus increasing system's infrastructure costs. SAIL opts for a different approach, since it uses a single AP together with users' smartphones sensors to estimate their position.

2.2.16 Pedestrian Dead Reckoning

Pedestrian Dead Reckoning is a positioning system that simply detects and estimates users' step length and direction of walking or heading, using DR and TOA location techniques by using users' device sensors and GPS positioning.

To detect and estimate the step length, Pedestrian Dead Reckoning starts by calculating the acceleration magnitude signal from the three orthogonal accelerometer signals from users' smartphone. Next, numerical step features are created based on the integral of the acceleration magnitude, acceleration's maximum and minimum values and variance. The numerical step features previously calculated are then used to train a neural network [19] to make step length predictions for relative positioning. The

output training patterns are the step lengths estimated from GPS position fixes, interpolated to footfall occurrences.

To detect and estimate the heading, a combination of magnetic compass measurements and GPS course over ground are used individually.

Results obtained have shown that the attained accuracy of the Pedestrian Dead Reckoning positioning approach is within 10 m when assuming that the distance between absolute GPS location fixes is less than about 500 m. Since Pedestrian Dead Reckoning applies DR techniques to locate users, which suffer from the accumulation of errors, it uses GPS to periodically fix those errors. However, GPS is not a viable approach for indoors, meaning that errors might not be efficiently corrected, which degrade system's accuracy and preclude it's use in such scenarios. Nevertheless, it collects accurate location information with low energy and infrastructure costs, since it only requires a smartphone with GPS, accelerometer, gyroscope and magnetometer.

2.2.17 UnLoc

UnLoc is an indoor positioning system that uses specific indoor locations, e.g an elevator or a corridor-corner as natural reference points". They consider that these locations might present unique signatures within the indoor space, thus allowing UnLoc to locate users by proximity when they reach those reference points, and by DR techniques between the same. First, mobile users move naturally between the indoor environment, collecting accelerometer, compass, gyroscope and WiFi readings. By assimilating data from these devices, UnLoc detects sensory signatures, e.g a corridor turn, that are unique within their respective WiFi sub-spaces. Then, using the collected data, UnLoc dead-reckons the devices starting from known reference locations, e.g the corridor turn. Results from 3 different indoor settings demonstrate median location error of 1.69 m.

They key UnLoc advantages are: i) it does not requires any pre-deployment efforts, since users only have to walk in the environment collecting data for UnLoc, and ii) it reduces the number of APs required to obtain users' location information, since it uses specific environmental characteristics as reference points.

Since UnLoc applies DR techniques, which suffer from the accumulation of errors, one of the main challenges was to deal with these errors that degrade system's accuracy. For outdoors, periodic recalibration with the GPS has been used [7]. However, GPS is not a viable approach for indoors. To replace GPS, UnLoc uses indoor landmarks which does not require any calibration efforts and helps reducing energy consumption, since GPS is a power hungry solution.

For outdoors, this system is not viable because natural landmarks are often nonexistent, e.g in a forest.

2.3 Comparative Study

Now that we have described in a broad manner, the main technologies and/or techniques used for localization and, in more detail, the positioning systems that implement them to collect location information, we present a comparative study of these systems (see Table 2.1), enhancing their results in terms of accuracy, energy consumption and infrastructure costs. The idea is to understand their differences to determine which should be supported in MultiTrack. The values provided in terms of energy consumption and infrastructure costs are classified using a five-point scale, where 1 is poor and 5 is excellent. We decided to use this scale because it allows us to easily compare the systems even when specific values of energy and infrastructure costs are not available.

System	Signal	Technique	Accuracy	Energy	Infrastructure	Focus
RADAR	RF	Fingerprint	3m	2	3	Indoor
Horus	RF	Fingerprint	0.6m	2	2	Indoor
EZ	RF	RSS	2m	2	4	Indoor
Labelee	RF	Proximity	-	5	2	Indoor
QR-Maps	-	Proximity	-	5	2	Indoor
Active Badge	IR	TOA	7cm	4	2	Indoor
Place Lab	RF	Proximity	20-30m	2	4	Indoor
RFIDLocator	RF	Proximity	-	3	2	Indoor
ZONITH	RF	RSS	2m	5	2	Indoor
LANDMARC	RF	RSS	1-2m	3	3	Indoor
TELIAMADE	Ultrasound	TOA	10cm	3	2	Indoor
Active Bat	Ultrasound	TOA	9cm	3	2	Indoor
Cricket	Ultrasound/RF	TOA	2cm	3	3	Indoor
GPS	RF	TOA	1-10m	1	5	Outdoor
SAIL	RF	TOA/DR	2-3m	2	4	Indoor
Pedestrian DR	RF	TOA/DR	10m	3	4	Outdoor
UnLoc	RF	Proximity/DR	2m	4	3	Indoor

Table 2.1: Overview of the positioning systems studied.

Table 2.1 shows that there are many different approaches when it comes to localization and all have different characteristics in terms of accuracy, energy and infrastructure costs that depend on the location technologies and/or techniques applied and scenarios where they are used. There isn't a single approach capable of acquiring location information with higher accuracy and lower costs than all the others in all possible scenarios. Moreover, all the positioning systems studied focus on particular scenarios and requirements by applying specific location technologies and/or techniques that best fit those cases. At last, Table 2.1 shows that systems with higher accuracy, tend to require higher infrastructure costs, e.g. Horus and Active Bat, and are used essentially indoors where the tracking areas are smaller to reduce the infrastructure costs. On the other hand, systems with lower accuracy require lower

infrastructure costs and can be used outdoors, e.g. GPS and Pedestrian DR.

Considering that the Cycle-to-Shop rewarding application is an example of a common LBSs and will be used to test MultiTrack, we need to take its requirements into account when it comes to choosing the location technologies and/or techniques that should be supported in MultiTrack. Having this into account, we need to support location technologies and/or techniques capable of providing location information with an accuracy on average better than 10 m, both indoors and outdoors and with minimal energy and infrastructure costs. For indoors, several approaches arise, e.g. Wi-Fi, RFID, NFC, IR, Ultrasound, Bluetooth, QR-codes and mobile sensors.

Wi-Fi is a valuable approach for scenarios where Wi-Fi networks are available, usually urban indoors, because it is capable of collecting location information with an accuracy better than 5 m, reasonable power consumption and without additional infrastructure costs, if the infrastructure required is already deployed, which is an often situation considering that it uses the infrastructure required by many other services, including Internet and IP telephony. Moreover, Wi-Fi is usually available on the mobile devices that run LBSs, which means that no additional costs are required in order to use this technology.

On the other hand, RFID, NFC, IR and Ultrasound require specialized infrastructure and are not usually available on the mobile devices that run LBSs, which increases the infrastructure costs required to use them.

Bluetooth and QR-codes also require specialized infrastructure, e.g. BLE beacons and codes, but they are usually available on the mobile devices that run LBSs, which means that users will not have additional costs in order to use these technologies for localization. For many indoor scenarios, they are valuable approaches. For example, Bluetooth beacons are a very interesting solution for scenarios where Wi-Fi networks are not available, because despite the higher infrastructure costs related with acquiring, deploying and maintaining the beacons, they can collect location information with similar accuracy to Wi-Fi, i.e. 5 m, with increased power savings, which is very useful for small indoor scenarios. On the other hand, QR-codes are an interesting solution for indoor scenarios where networks are not available and energy and infrastructure costs need to be minimal.

At last, the usage of Mobile sensors is mandatory because they are part of mobile devices and can be used to complement the location information collected by other solutions, as we have seen in SAIL, Pedestrian DR and UnLoc.

For outdoors, GPS is clearly the best approach, despite the high energy consumption associated. It collects relatively accurate location information (1-10 m) with no additional infrastructure costs, since it is usually available on mobile devices that run LBSs.

Taking this into account, we can conclude that GPS, Wi-Fi and mobile sensors are the location technologies and/or techniques that best fit the majority of LBSs requirements and due to that, should be first supported in MultiTrack. Moreover, for indoor scenarios where Wi-Fi networks are unavailable,

the location information collected using GPS and mobile sensors might not achieve Cycle-to-Shop's requirements, as shown by Pedestrian DR [7]. For this case, we decided to also support the usage of Bluetooth, since it provides location information with high accuracy (2 m) and low power consumption and Cycle-to-Shop already assumes the existence of these beacons inside partner's stores.

Nevertheless, this does not mean that the remaining location technologies and/or techniques should be discarded from MultiTrack. As we have seen, all location technologies and/or techniques studied have different characteristics, advantages and disadvantages, and so, all are valuable solutions from providing location information for LBSs. It all depends on LBSs requirements, e.g. an LBS which requires maximum energy savings or accuracy might require the usage of QR-codes or Ultrasound to acquire the location information that best fit its requirements.

In the next Chapter, we describe the MultiTrack architecture, focusing on its main components and interactions, enhancing how they support the flexible usage of different location technologies and/or techniques by LBSs, including GPSs, Wi-Fi, mobile sensors and Bluetooth, and ease the addition of new ones.

3

Architecture

Contents

In this Chapter, we present the MultiTrack architecture (see Figure 3.1), which can be broken into the following components: i) the Tracker, ii) the Persistent Track Storage, and iii) the Client (singleton). In the next sections, we describe these components and their interactions (including those with external entities, e.g. the Mobile Hardware), enhancing how they achieve the architectural design requirements.

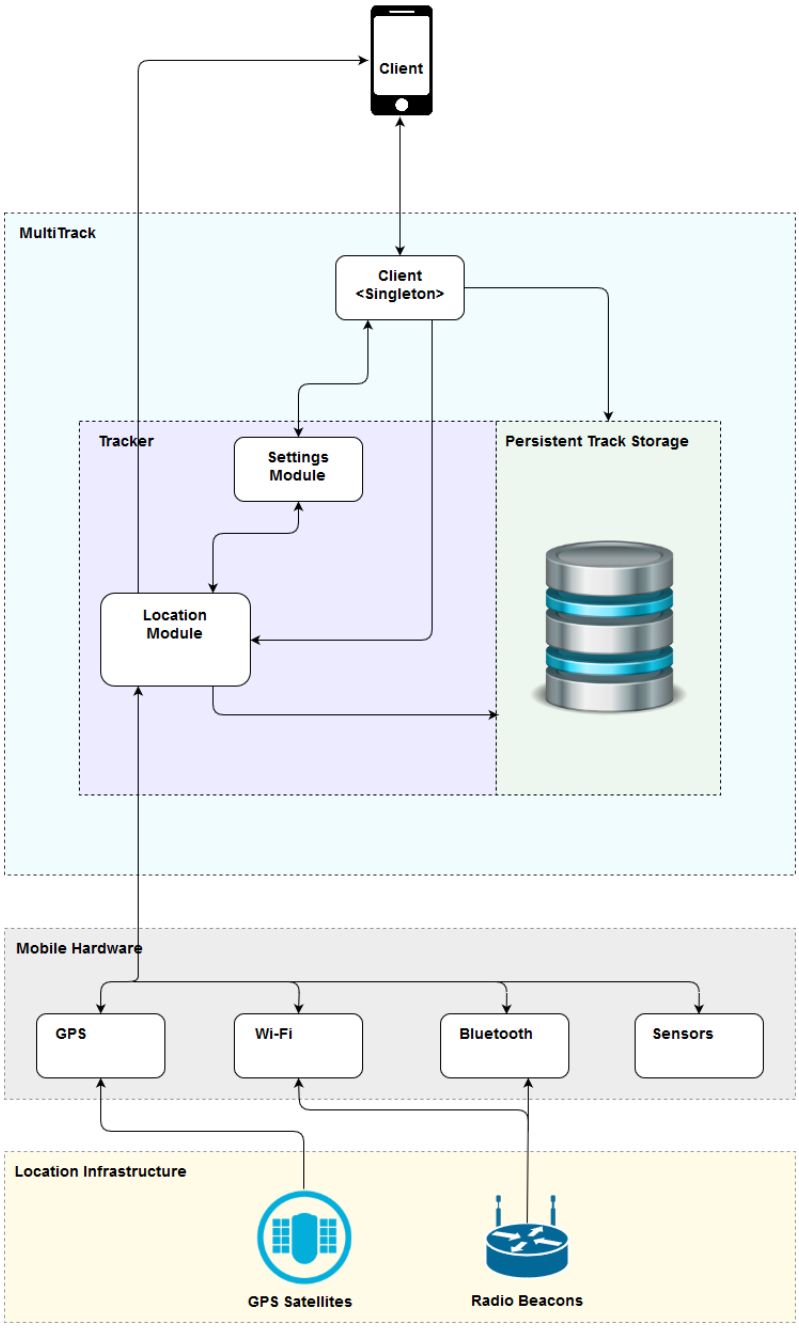

Figure 3.1: The MultiTrack architecture.

3.1 Tracker

The Tracker is responsible for collecting location information from available sources, e.g. GPS, Wi-Fi, Bluetooth, QR-codes, NFC and mobile sensors (accelerometer, gyroscope and magnetometer), and for delivering that information to clients, i.e. LBSs. Moreover, it enables clients to control the process of collecting location information, by allowing them to choose the location technologies and/or techniques and corresponding configurations, used to acquire the location information. It has 2 modules: i) the Location Module, and ii) the Settings Module.

3.1.1 Location Module

The Location Module collects location information and shares it with clients. It contains several sub-modules, that collect location information using different location technologies and/or techniques. Moreover, considering the Modularity architectural design requirement, each of these sub-modules need to be easily added and removed. To achieve it, the Location module has an API (see Listing 3.1), that is implemented by each of its sub-modules.

Listing 3.1: Interface provided by the Location Module.

```
1
2 public interface LocationInterface {
3     void start();
4     void stop();
5 }
```

Using this approach, adding or removing location technologies and/or techniques to the Location Module, requires only the addition or removal of the corresponding sub-modules, instead of a redesign of the whole system. This allows MultiTrack to be easily updated as new location technologies and/or techniques are discovered, or sensors added to mobile devices.

3.1.2 Settings Module

The Settings module stores Location's sub-modules configurations and enables clients to control them, i.e. read and update them. These configurations are loaded by the Location module when its functioning is requested, and they specify which sub-modules are used to collect location information and their configurations, e.g. use GPS with 3 s rate. The usage of this module is crucial for achieving the Technologies Control architectural design requirement, since it enables clients to control Location's sub-modules, and consequently, the location technologies and/or techniques supported in MultiTrack.

Moreover, it contributes for achieving the Energy Efficiency architectural design requirement, since it enables clients to adjust the process of collecting location information to best fit their requirements, thus increasing power savings.

Clients can control the configurations stored at the Settings module by using the Client (singleton), which will be described further. However, the impact that these configurations have on Location's sub-modules has limitations, e.g. changing configurations during the Location module functioning will not affect the ongoing collection of location information, only the next, since these configurations are loaded by the Location module only when it starts functioning.

When the Location Module is requested to start collecting location information, it creates an high-level entity called Track, to encapsulate all the acquired location information, e.g. users' positions, velocity, duration and distance. Then, the Location module loads the configurations stored at the Settings module and starts the corresponding sub-modules with those configurations. The sub-modules will then communicate with the mobile hardware to start collecting location information, which send the acquired information back to the corresponding sub-modules. At last, the sub-modules add the information received to the current Track. When the Location module is requested to stop collecting location information, it communicates with the working sub-modules to stop the corresponding mobile hardware modules, and then, the Track is directly sent to the client, or stored at the Persistent Track Storage to be persistently stored at mobile's device local memory. Whether the Track is directly sent to the client or stored at the Persistent Track Storage depends on the sub-modules and operations requested by the client to collect location information.

3.2 Persistent Track Storage

The Persistent Track Storage is responsible for storing the Location information collected by the Tracker, i.e. the Track, on mobile's device local memory, and for allowing clients to access it by using the Client (singleton). It applies the One-To-Many Relational Model by using 2 tables: i) the parent, which stores all the high-level information related with a Track, e.g. total distance and average speed, and ii) the child, which contains all the location information collected by the framework for a particular Track, i.e. user's positions. We opted by using this approach instead of a single table with all the data associated with a Track because it stores less repeated data and so, it decreases the amount of memory used.

At last, when the Persistent Track Storage receives a Track to be stored, it applies filters to discard repeated coordinates or coordinates with low accuracy. The objective is to reduce the amount of useless or erroneous information stored on mobile's local memory and the corresponding energy required to store it, thus contributing to achieve the Energy Efficiency architectural design requirement.

3.3 Client (singleton)

The Client (singleton) is responsible for allowing clients to communicate with the Tracker and Persistent Track Storage components. It is the clients' entry point to the framework, since it enables access to all operations supported, and it ensures that the framework has only one entry point by applying the Singleton Design Pattern. The operations available are the following:

- Get Client Instance, which fetches an instance of the Client (singleton).
- Start Tracking, which requests the Location module to start collection location information with the location technologies and/or techniques and corresponding configurations, stored at the Settings module. First, it creates a new Track, identified by an unique ID, that is used to group all the collected information. Then, the Location module loads the configurations stored at the Settings module, initiates its sub-modules with those configurations, and starts the requested sub-modules, which communicate with the corresponding mobile's hardware modules to start acquiring location information. The information acquired, is then sent directly to the sub-modules that store it in the Track.
- Stop Tracking, which requests the Location module to stop the working sub-modules and corresponding hardware modules. Moreover, it sends the Track with the collected location information to the Persistent Track Storage, or directly to the client, depending on the sub-modules requested.
- Get Last Location, which requests the Location module to obtain the last known location of users' device using the requested sub-modules. Then, it sends this information directly to the client.
- Get Settings, which retrieves the configurations stored at the Settings module.
- Update Settings, which allows clients to update the configurations stored at the Settings module.
- Get Stored Tracks, which fetches the list of all Tracks stored at the Persistent Track Storage. Each track contains top-level information, such as the elapsed time and traveled distance.
- Get Stored Track, which fetches the Track identified by its unique ID from the Persistent Track Storage.
- Delete Store Track, which deletes the Track identified by its unique ID from the Persistent Track Storage.

Now that we have described the MultiTrack architecture, focusing on its main components and interactions, lets consider the hypothetical scenario of a client that wants to use MultiTrack to collect location information using different location technologies and/or techniques, e.g. GPS, Wi-Fi, Bluetooth and mobile sensors. First, the client establishes a connection with the MultiTrack service to obtain an instance

of the Client (singleton). Then, the client uses that instance to call the Update Settings operation and choose the location technologies and/or techniques used by MultiTrack to collect location information, i.e. GPS, Wi-Fi, Bluetooth and mobile sensors. Then, the client calls the Start Tracking operation and MultiTrack creates a new Track and initiates its sub-modules with the previously defined configurations, starting those required by the client (see Listing 3.2).

Listing 3.2: Example of a client establishing a connection with the MultiTrack service in order to use it.

```
1
2 connection = connectTo(multiTrackService);
3 client = connection.getClientInstance();
4 client.updateSettings();
5 client.startTracking();
```

At this point, the requested sub-modules will communicate with the corresponding mobile hardware modules, i.e. GPS, Wi-Fi, Bluetooth and mobile sensors (accelerometer, gyroscope and magnetometer) to start collecting location information. At last, the collected information is sent back to the sub-modules, which store it at the current Track.

To stop the collection of location information, the client calls the Stop Tracking operation, which tells MultiTrack to stop its services, i.e. the working sub-modules and corresponding mobile hardware modules. At last, the current Track can be either stored at the Persistent Track Storage or sent directly to the client.

As we can see, after establishing a connection with the MultiTrack service and consequently, gain access to the Client (singleton), clients can easily control framework's operations and take advantage of the supported location technologies and/or techniques to collect location information.

In the next Chapter, we describe the options available and the choices made for implementing the location technologies and/or techniques chosen, as well as the core components of MultiTrack, enhancing their advantages and disadvantages at the light of MultiTrack's requirements.

4

Implementation

Contents

Before implementation begins, several questions arise: i) For which platforms should MultiTrack be developed?, ii) What location technologies and/or techniques should be supported in the Location Module and which APIs should be used to implement them?, iii) What configurations should be available at the Settings Module?, and iv) How will the Persistent Track Storage store the Tracks?

In the next sections, we answer to these questions, describing the options available and the choices made, enhancing their advantages and disadvantages at the light of our requirements.

4.1 Platform

There are several platforms powering mobile smartphones today, e.g. Android, iOS, Windows 10 mobile and Blackberry. For large companies, with many resources, development can be done simultaneously for different platforms, thus reaching a larger number of users, but for solo developers and small companies (our case), with few resources, development usually starts for a specific platform, and then, it is expanded to others if required. Taking this into account, our objective is to choose from these platforms, the one that allows MultiTrack to reach more users with less development costs.

According to IDC[1], the 2 mobile platforms with more smartphones shipped worldwide are Android and iOS. Android dominates the number of smartphones shipped worldwide with a market share of approximately 80% and iOS comes next with a market share of approximately 16%. In terms of development costs, Android is less expansive than iOS, since to develop for iOS, a developer must use a Mac and register on the Apple App Store, which requires a yearly fee of 99$, whereas to develop for Android, it can be done on Windows, Mac or Linux and requires a one time payment of 25$ to register on the Google Play Store. Having this into account, we opted for starting developing MultiTrack for the Android platform, since out objective is to reach the maximum number of users with the lowest development costs.

4.2 Location Technologies and APIs

As already said on Chapter 2, the location technologies and/or techniques that must be initially supported in MultiTrack are GPS, Wi-Fi, Bluetooth and mobile sensors (accelerometer, gyroscope and magnetometer). To implement these location technologies and/or techniques for the Android platform, there are several APIs available. In the next subsections, we describe the options available and the choices made, enhancing their advantages and disadvantages at the light of our requirements.

[1] http://www.idc.com/prodserv/smartphone-os-market-share.jsp

4.2.1 GPS, Wi-Fi and Mobile Sensors

In Android, the 2 most relevant location APIs that use GPS and Wi-Fi are the android.location[2] and the Google Location Services[3].

The android.location API uses GPS and Wi-Fi to collect location information and requests clients to manually choose the providers to use. On the other hand, the Google Location Services API uses GPS, Wi-Fi and mobile sensors through the Fused Location Provider API, to collect location information. Also, it chooses automatically, based on the accuracy of the information, battery usage and priority, the providers to use at a moment, and merges the data collected by these providers to obtain the most accurate location information.

There are 4 different priorities available that are manageable by clients: i) High Accuracy, ii) Balanced Power Accuracy, iii) Low Power, and iv) No Power. These priorities specify the focus of the API, relative to accuracy and energy consumption. Setting the priority to High Accuracy configures the API to focus on collecting the most accurate location information, no matter the energy costs associated. For example, when users are outdoors, the API uses essentially GPS, while indoors, it uses Wi-Fi and mobile sensors[4]. Setting the priority to No Power configures the framework to focus on increasing energy savings, i.e. to collect location information with lower accuracy and less often. For example, it might stop using GPS or reduce dramatically its rate to save energy, since GPS is a power consuming solution, as described on Chapter 2. By allowing clients to control the priority used, the API allows them to adjust the collection of location information to better fit their requirements.

Another API that is part of the Google Location Services API is the Activity Recognition API. It uses the same providers as the Fused Location Provider API but to detect users' activity. It periodically wakes up the device and reads short bursts of sensor data from low power sensors, in order to keep the power usage to a minimum. It is capable of detecting if users are currently on foot, in a car, on a bicycle or still with a determined confidence level, which is a value from 0 to 100 indicating the likelihood that users are actually performing such activity. Moreover, it stops when detects that users are still, thus increasing power savings.

Actually, there are several LBSs that require users' activity detection in order to provide their services. Cycle-to-Shop is an example of such LBSs, since users are supposed to be rewarded only if they go to stores while cycling. Another example of a LBS that requires users' activity detection to provide its services is Pokémon GO[5], where users need to walk a determined distance in order to hatch Pokémon eggs. If users are not walking, instead they are driving or in a train, the distance is not updated and the eggs do not hatch.

[2] https://developer.android.com/reference/android/location/package-summary.html
[3] https://developers.google.com/android/reference/com/google/android/gms/location/package-summary
[4] Google I/O 2013 - Beyond the Blue Dot: New Features in Android Location
[5] http://www.pokemongo.com/

Taking into account that the Google Location Services API is: i) the preferred way to add location-awareness for Android applications, ii) contains the Fused Location Provider API, which chooses automatically, based on the accuracy of the information, battery usage and priority, the providers to use at a moment, and merges the data collected to obtain location information more accurate and with lower energy costs than the android.location API, and iii) contains the Activity Recognition API, which is able to detect user's activity, we opted by using it in MultiTrack, rather than the android.location API.

4.2.2 Bluetooth

Regarding the usage of Bluetooth Beacons in Android, there are several solutions that are at our disposal, e.g. Blueup[6], Kontakt[7] and Estimote[8]. Our intention is to support the maximum number of Bluetooth beacons, thus enabling LBS to take advantage of these infrastructures more often, but since, there isn't a single API that allow us to detect and communicate with all beacons, despite their similarities, and this is a project with time constraints, we had to make a choice. We opted by using the Estimote Beacons in MultiTrack because they have a simpler and more intuitive API than the remaining solutions. Nevertheless, the Location module is designed to ease the addition and removal of location APIs.

The API that supports the usage of Estimote Beacons for localization purposes is called Estimote API. This API enables LBSs to detect Bluetooth Estimote beacons nearby, and consequently, determine users' position by proximity to them. Estimote beacons are small wireless devices with an unique ID that are continuously broadcasting BLE signals, which are received and interpreted by the Estimote API, thus enabling contextual awareness. When MultiTrack detects one of these signals, it sends a message containing the ID of the beacon to the LBS, allowing it to determine users' position by proximity to the beacon with high accuracy and low energy costs, since the LBS knows the IDs and corresponding locations of the beacons.

4.3 Client's Configurations

The Settings Module stores Location's sub-modules configurations (see Figures 5.1 and 5.2) and allows clients to control them, by using the operations available at the Client (singleton). Regarding the Fused Location Provider API, there are 5 configurations stored at the Settings Module that are manageable by clients: i) start/stop, ii) Interval, iv) Fastest Interval, and v) Priority.

The start/stop determines if the Location sub-module requests the usage of the Fused Location Provider API to collect location information.

[6] http://www.blueupbeacons.com/
[7] https://kontakt.io/
[8] http://estimote.com/

The Interval sets the time in milliseconds between location updates. The Fastest Interval sets a limit for the Interval parameter, since several LBSs might be requesting location updates at different rates. Decreasing this value will increase the number of location updates and consequently the energy consumed.

The Priority establishes the focus of the location request, i.e the balance between accuracy and energy consumption. If set it to ”High Accuracy”, the Fused Location Provider API will use all location providers available, i.e. GPS, Wi-Fi and mobile sensors, and combine them to collect the most accurate location information, but with the highest costs in terms of energy consumption. On the other hand, if set to ”No Power”, the Fused Location Provider API will will acquire location information less accurate, but also with less costs in terms of energy consumption.

Regarding the Activity Recognition API, there are 2 configurations stored at the Settings Module that are manageable by clients: i) start/stop, ii) Interval, and iii) Confidence.

The start/stop determines if the Location sub-module requests the usage of the Activity Recognition API to collect location information. The Interval sets the time in milliseconds between activities detection. Larger values will result in fewer activity detections, thus reducing energy consumption, while smaller values will result in more frequent activity detections, which increases power consumption. The Confidence establishes the minimum acceptable confidence for results provided by the Activity Recognition API to be taken into account by the framework. Activities detected with a confidence level lower than this value are discarded.

Regarding the Estimote API, there is only the start/stop setting, which as we have seen before, determines if the Location module requests the usage of the Estimote API to collect location information.

4.4 Track Storage

Android provides several options to save persistent application data on mobile’s device local memory: i) Shared Preferences, ii) Internal Storage, iii) External Storage, and iv) SQLite Databases. Choosing between these approaches depends on application’s requirements, e.g. type of data to be stored, space available and access restrictions. In the next subsections, we describe these approaches, enhancing their advantages and disadvantages, to choose which one to use at the Persistent Track Storage to store the Tracks.

4.4.1 Shared Preferences

The Shared Preferences approach allows to save and retrieve data from persistent key-value pairs. The data stored is limited to the primitive data types, e.g booleans, floats, integers, longs and strings, and is private to the application that stored it, meaning that other applications cannot access it, nor

can the user. Moreover, the data persists even when the application is killed. The Shared Preferences approach is ideal for storing simple application's data, e.g. configurations, since it stores only primitive data types and the information is private to the application that stored it.

4.4.2 Internal Storage

The Internal Storage approach allows to save and retrieve data from files stored at device's internal memory. The data can be of any type, is private to the application that stored it, and is only discarded when the application is uninstalled. The Internal Storage approach is very similar to the Shared Preferences, with the difference that allows to save all types of data, instead of primitive types only. It is ideal for storing private complex application's data since it supports the storage of all types of data and the information is private to the application that stored it.

4.4.3 External Storage

The External Storage approach allows to save and retrieve data from files stored at an external storage, e.g. an SD card. The data can be of any type, is world-readable and modifiable by the user, i.e. all application can read and write the files placed on the external storage and the user can remove them. This type of storage is often used to store songs, movies and images that are meant to be accessed by other applications and by users.

4.4.4 SQLite Databases

The SQLite Databases approach allows to save and retrieve data from SQLite databases stored at device's internal memory. The data can be of any type, is private to the application that stored it, and is only discarded when the application is uninstalled.

One of the major advantages of the SQLite databases is that its file format is cross-platform. A database file written on one machine can be copied to and used on a different machine with a different architecture. This type of storage is often used to store private repeating or structured data, e.g. contact information or location information[9].

Considering that a Track is composed by users' position, velocity, duration and distance, i.e. structured and repeating information that should be private (only the application that stored it should be able to access it), there are 2 approaches available in Android that we can use to implement the Persistent Track Storage: i) Internal Storage, and ii) SQLite databases. Taking into account that SQLite databases are ideal for storing repeating or structured data, such as Tracks, and that they offer cross-platform sup-

[9] https://developer.android.com/training/basics/data-storage/databases.html

port, thus easing the development of MultiTrack for other mobile platforms, we opted by using SQLite databases to implement the Persistent Track Storage.

In the next Chapter, we describe the experiments done using the Cycle-to-Shop rewarding application with MultiTrack to collect location information. The objective of these experiments is to determine if MultiTrack achieves the goal of supporting the flexible usage of different location technologies and/or techniques by mobile smartphone LBSs. At last, we present the experimental results obtained and corresponding conclusions, enhancing the importance of supporting different location technologies and/or techniques and allowing LBSs to control them, to the collection of the location information that best fit their requirements.

5

Experimental Results

Contents

To test if MultiTrack supports the flexible usage of different location technologies and/or techniques by Android smartphone LBSs, we decided to develop an online Android smartphone LBS called Cycle-to-Shop, that requests MultiTrack to collect location information using GPS, Wi-Fi, mobile sensors (accelerometer, gyroscope and magnetometer) and Bluetooth.

In this Chapter, we describe the experiments done using the Cycle-to-Shop rewarding application with MultiTrack to collect location information. The objective of these experiments is to determine if MultiTrack achieves the requirements presented in Chapter 1 and consequently, the goal of supporting the flexible usage of different location technologies and/or techniques by mobile smartphone LBSs. At last, we present the experimental results obtained and corresponding conclusions.

5.1 Experiments

To test the framework, the following experiments were done: i) the Accuracy Experiment, ii) the Costs Experiment, and iii) the User's Activity Detection Experiment.

These experiments were conducted in a LG K8 4G Android API 6.0 running the Cycle-to-Shop rewarding application and requesting MultiTrack to provide the required location information. Moreover, 2 different configurations for MultiTrack were used during experiments, called High Accuracy and Low Power. The High Accuracy configuration sets the priority parameter to "High Accuracy", the interval to 3500 ms, the Fast Interval to 3000 ms and disables user's activity detection (see Figure 5.1). The Low Power configuration only changes the priority parameter to "Low Power" relative to the High Accuracy configuration (see Figure 5.2).

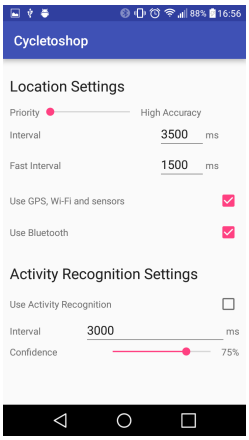

Figure 5.1: Cycle-to-Shop configuring MultiTrack for High Accuracy.

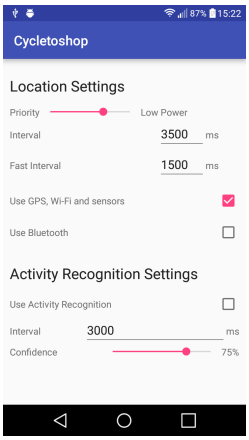

Figure 5.2: Cycle-to-Shop configuring MultiTrack for Low Power.

Despite these configurations influence only the behavior of the Fused Location Provider API, since the Estimote API can only be started and stopped by clients, they allow us to test client's control over the

location technologies and/or techniques supported in MultiTrack, and how that influences the information collected.

5.1.1 Accuracy Experiment

The objective of the Accuracy Experiment is twofold: i) test the accuracy of the location information collected by MultiTrack to determine if the framework achieves Cycle-to-Shop's accuracy requirements, i.e. an accuracy less than 10 m both indoors and outdoors, and ii) test if MultiTrack achieves the Technologies Control requirement presented on Chapter 1.

The experiment consists on a user walking in circles 3 m away from store's coordinates during 5 minutes. We have chosen these conditions because they allow us to determine the behavior of the location technologies and/or techniques used by MultiTrack in most scenarios.

The experiment was made in 2 different scenarios, one indoors and the other outdoors, both with Wi-Fi availability (see Figures 5.3 and 5.4).

During the experiment, the user is running the Cycle-to-Shop rewarding application, which is requesting MultiTrack to collect location information using either, the Fused Location Provider API (configured for High Accuracy and Low Power) and the Estimote API. The information acquired by MultiTrack is sent to Cycle-to-Shop, allowing it to calculate the distance between the user and store's coordinates, and compare that value with the real distance, i.e. 3 m, to determine the accuracy of the framework in each scenario.

Figure 5.3: Indoors scenario with an Wi-Fi AP and without obstacles between the user and store's coordinates.

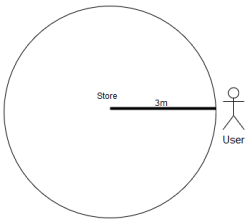

Figure 5.4: Outdoors scenario without obstacles between the user and store's coordinates.

During the experiment, the user is constantly moving because the Fused Location Provider API uses mobile sensors, and when these sensors detect that the user is still, the API does not re-calculate user's position, instead it returns a previous calculation to increase energy savings. To trick mobile sensors, the user is constantly moving in a circle around store's coordinates, thus maintaining its distance to the store but forcing the API to re-calculate user's position.

5.1.2 Costs Experiment

The objective of the Costs Experiment is to determine the energy and infrastructure costs required by MultiTrack to collect location information, to test if it achieves the Energy Efficiency requirement presented on Chapter 1.

The experiment consists on a user driving a car through an urban outdoors route (see Figure 5.5) with 14.2 km distance according to Google Maps[1], running the Cycle-to-Shop rewarding application. The experiment has the duration of 25 minutes and during this time, Cycle-to-Shop is requesting MultiTrack to collect location information using either, the Fused Location Provider API (configured for High Accuracy and Low Power) and the Estimote API. The experiment was repeated 10 times for each configuration used in MultiTrack. We opted by using a car to do this experiment because the route is very long and the experiment is repeated several times.

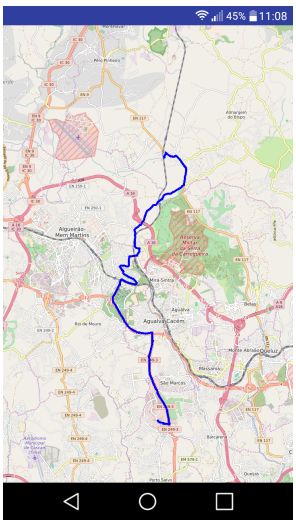

Figure 5.5: Urban outdoors route with 14.2 km distance according to Google Maps.

5.1.3 User's Activity Detection Experiment

The objective of the User's Activity Detection Experiment is to test if MultiTrack correctly detects user's modality.

The experiment consists on a user passing in front of a partner's store while cycling. The user is running the Cycle-To-Shop rewarding application, which is requesting MultiTrack to collect location information using the Activity Recognition API. The experiment was repeated 15 times. We focused only on this particular modality because it is the most important taking into account Cycle-to-Shop's requirements, since it needs to detect when users are cycling in order to reward them accordingly.

[1] https://www.google.pt/maps

5.2 Results

In this section, we present for each experiment, the results obtained by MultiTrack using the Estimote API and the Fused Location Provider API configured for High Accuracy and Low Power.

5.2.1 Accuracy Experiment

Results obtained indoors during the Accuracy Experiment (see Figure 5.6) show us that the Fused Location Provider API configured for High Accuracy is, on average, more accurate than the Fused Location Provider API configured for Low Power and the Estimote API. The mean values of the distances between the user and store's coordinates obtained by MultiTrack using the Fused Location Provider API configured for High Accuracy and Low Power are of 3.78 m and 20.27 m respectively, while the mean value of the Estimote API is 5.82 m. Considering that the user is always at 3 m from store's coordinates, MultiTrack using the Fused Location Provider API configured for High Accuracy and Low Power shows an average accuracy of 0.78 m and 17.27 m respectively, while using MultiTrack with the Estimote API shows an accuracy of 2.82 m.

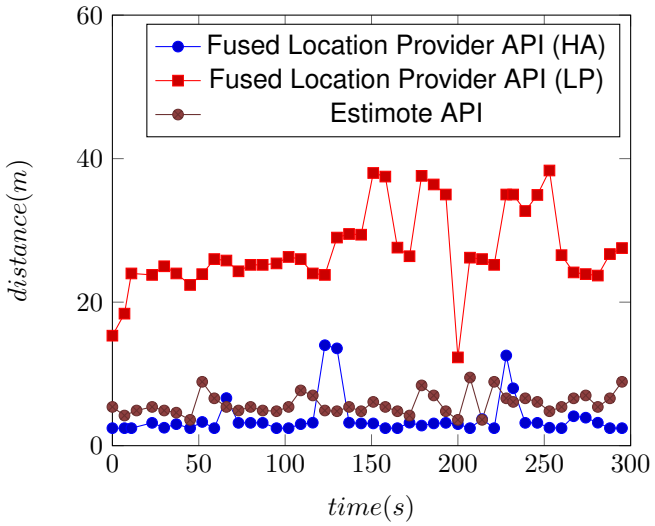

Figure 5.6: Distance in meters between the user and store's coordinates acquired by MultiTrack indoors using either the Fused Location Provider API, configured both for High Accuracy (HA) and Low Power (LP), and the Estimote API.

Results obtained outdoors (see Figure 5.7) show us that, once again, the Fused Location Provider API configured for High Accuracy is, on average, more accurate than the Fused Location Provider API configured for Low Power and the Estimote API. The mean values of the distances between the user and store's coordinates obtained by MultiTrack using The Fused Location Provider API configured for High Accuracy and Low Power are of 4.63 m and 29.29 m respectively, while the mean value of the Estimote API is 5.9 m. Considering that the user is always at 3 m from store's coordinates, MultiTrack using the

Fused Location Provider API configured for High Accuracy and Low Power shows an average accuracy of 1.63 m and 26.29 m respectively, while using MultiTrack with the Estimote API shows an accuracy of 2.9 m. The reason why the Fused Location Provider API configured for Low Power has such a high error when compared to the remaining approaches is because it focuses on increasing power savings, and so it does request the usage of all mobiles' available sensors to acquire location information.

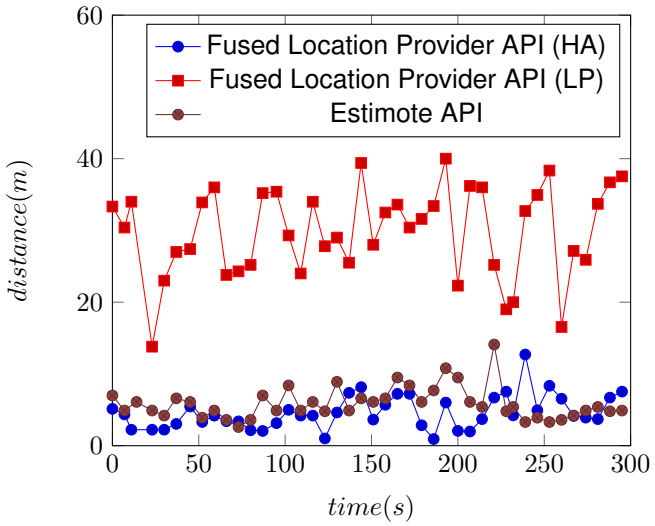

Figure 5.7: Distance in meters between the user and store's coordinates acquired by MultiTrack outdoor using either the Fused Location Provider API, configured both for High Accuracy (HA) and Low Power (LP), and the Estimote API.

5.2.2 Costs Experiment

Results obtained during the Costs Experiment (see Figure 5.8) show us that, in terms of energy consumption, the Fused Location Provider API is far more energy hungry than the Estimote API, specially when configured for High Accuracy. The percentage of device's battery consumed by the application using MultiTrack with the Fused Location Provider API configured for High Accuracy and Low Power is approximately 5 % and 2 % respectively, while the percentage of device's battery consumed by the application using MultiTrack with the Estimote API is approximately 1 %.

In terms of infrastructure costs, the Estimote API is far more expensive than the Fused Location Provider API, independently from the configuration used, because it uses Estimote beacons in addition to the mobile's hardware, i.e. Bluetooth sensors, while the Fused Location Provider API uses only mobile's hardware, i.e. GPS, Wi-Fi and mobile sensors (accelerometer, gyroscope and magnetometer). Each Estimote beacon has a cost of 20 dollars and a lifetime and range of approximately 1 year and 70 m respectively, meaning that for tracking an user through the route shown on Figure 5.5, at least 202 beacons need to be acquired and deployed representing a total cost of approximately 4000 dollars,

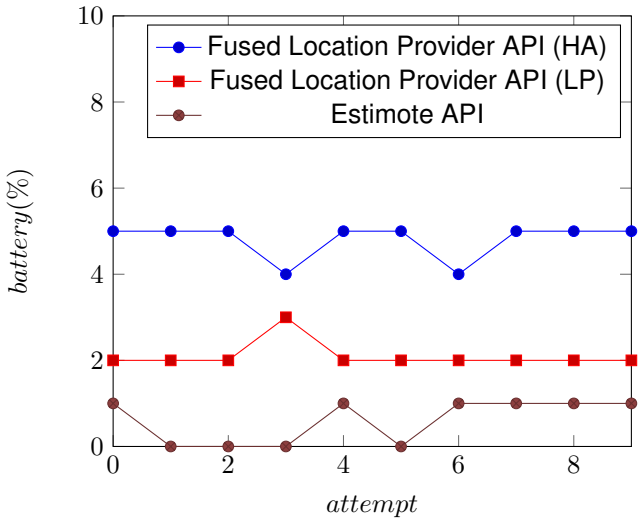

Figure 5.8: Percentage of device's battery spent by the application using MultiTrack with the Fused Location Provider API, configured both for High Accuracy (HA) and Low Power (LP), and the Estimote API.

excluding maintenance costs. On the other hand, tracking an user using the Fused Location Provider API has no additional costs to those required to acquire the mobile device.

5.2.3 User's Activity Detection Experiment

Results obtained during the User's Activity Experiment (see Figure 5.9) show us that MultiTrack successfully detected when the user passed nearby the store while cycling.

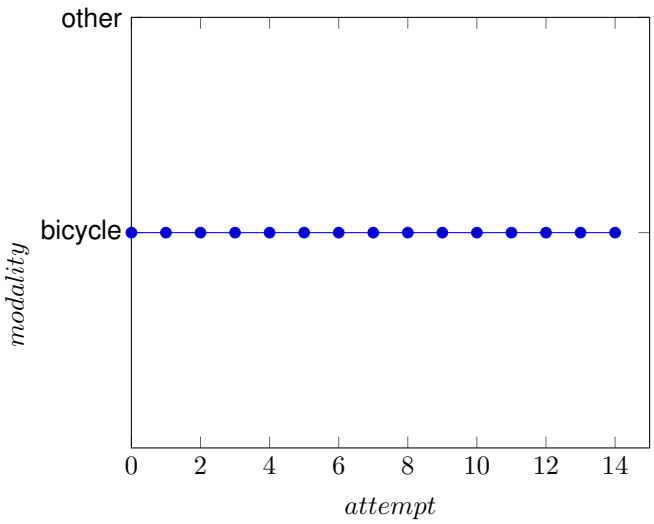

Figure 5.9: User's modality collected by MultiTrack using the Activity Recognition API.

5.3 Experimental Conclusions

Results obtained during the Accuracy Experiment show that on average, MultiTrack has an accuracy less than 10 m, both indoors and outdoors, using either the Fused Location Provider API configured for High Accuracy and the Estimote API, thus achieving Cycle-to-Shop's accuracy requirements.

Results obtained during the Costs Experiment show that MultiTrack consumes less energy using the Estimote API than using the Fused Location Provider API. However, in terms of infrastructure costs, MultiTrack has higher costs using the Estimote API than using the Fused Location Provider API, specially when tracking users in large areas. This shows that there is a trade-off between these APIs when it comes to energy consumption vs infrastructure costs.

Results obtained during the User's Activity Detection Experiment show that MultiTrack successfully detects when users are cycling.

Taking these results into account, we can conclude that MultiTrack successfully achieves its goal of supporting the flexible usage of different location technologies and/or techniques by mobile smartphone LBSs because: i) it achieves the Modularity architectural design requirement by requiring the implementation of a single API (LocationInterface) for all location technologies and/or techniques supported, as described on Chapter 3, ii) it achieves the Technologies Control architectural design requirement by allowing LBSs to control the supported location technologies and/or techniques, as also described on Chapter 3 and shown during Cycle-to-Shop's experiments, and iii) it achieves the Energy Efficiency architectural design requirement by supporting the usage of energy efficient location APIs, e.g. Fused Location Provider API and Estimote API.

Moreover, these results also show that MultiTrack achieves Cycle-to-Shop's requirements by supporting the flexible usage of different location technologies and/or techniques since: i) on average it provides location information with an accuracy less than 10 m, both indoors and outdoors, using different location technologies and/or techniques, ii) it successfully detects when users are cycling, iii) it minimizes energy costs by supporting the flexible usage of energy efficient location APIs, e.g Fused Location Provider API and Estimote API, and iv) it minimizes infrastructure costs because it supports the flexible usage of the Fused Location Provider API, which requires no additional infrastructure costs to those required by the user to acquire the mobile device.

In the next Chapter, we present the conclusions inferred during the project and MultiTrack's limitations and future work.

6

Conclusion

Contents

In this Chapter, we present MultiTrack's conclusions, limitations and future work.

6.1 Conclusions

As we have seen on Chapter 2, there are many different approaches when it comes to localization and all have different characteristics in terms of accuracy, energy and infrastructure costs that depend on the location technologies and/or techniques applied and scenarios where they are used. There isn't a single approach capable of acquiring location information with higher accuracy and lower costs than all the others in all scenarios. Having this into account and considering that LBSs requirements may vary and are not known in advance, we cannot simply choose one approach to deliver the location information that LBSs require, i.e. that best fit their requirements. Instead we should use different location technologies and/or techniques and apply them at the environments where they achieve the best results, i.e. the highest accuracy with the lowest energy and infrastructure costs.

There are several positioning systems that already use this concept, e.g. Google Location Services, which merges the information collected by GPS, Wi-Fi and mobile sensors (accelerometer, gyroscope and magnetometer). However, all these approaches focus on specific location technologies and/or techniques, instead of trying to support them all. For this reason, we propose a different approach, called MultiTrack, which focus on supporting the flexible usage of all location technologies and/or techniques, so that LBSs can acquire the location information that best fit their requirements.

Experimental results prove that MultiTrack achieves Cycle-to-Shop's requirements by supporting the flexible usage of GPS, Wi-Fi, Bluetooth and mobile sensors, and they show us that allowing LBSs to control these techniques contributes for providing the information that they require, i.e. that best fit their requirements. Nevertheless, MultiTrack still has many limitations and future work, which we describe in the next subsection.

6.2 System Limitations and Future Work

There are several limitations and future work related with MultiTrack. First, it was only developed for the Android operating system, and considering that it aims at providing location information for all LBSs, not only the Android ones, it needs to be deployed for the remaining platforms.

Second, more location technologies and/or techniques should be supported in MultiTrack, e.g. QR-codes, RFID and so on. Considering that MultiTrack's goal is to collect the location information that best fit LBSs requirements, we cannot simply support the flexible usage of GPSs, Wi-Fi, Bluetooth and mobile sensors, because as we have seen on Chapter 2, the remaining location technologies and/or techniques collect location information with different characteristics, which can be useful for LBSs. Having this

into account, we can assume that this topic will be always a work in progress, since as new location technologies and/or techniques are being discovered or improved, they need to be added to MultiTrack.

At last, MultiTrack should be able to automatically change the location technologies and/or techniques used to collect location information, taking into account LBSs requirements and environmental characteristics. At the moment, LBSs have full control over MultiTrack's capabilities, i.e. they choose the location technologies and/or techniques and corresponding configurations used to collect location information, which contributes to align the location information to best fit their requirements. However, for cases where environments change unexpectedly, the location information can be highly affected. For example, lets imagine a LBS that needs to detect users inside a particular area covered by Bluetooth beacons and is requesting MultiTrack to collect location information using Bluetooth. If those beacons fail and stop broadcasting Bluetooth signals, MultiTrack wont be able to detect users nearby and the LBS will consequently fail to provide its services. In these cases, it would be interesting to detect the anomaly and automatically change the collection of location information to use a different approach, capable of acquiring location information closer to LBSs expectations.

In the remaining of this document we present the Bibliography and the Mockups used to build the Cycle-to-Shop rewarding application.

Bibliography

[1] P. Bahl and V. N. Padmanabhan, “Radar: An in-building rf-based user location and tracking system,” in *INFOCOM*, 2000, pp. 775–784.

[2] P. Misra and P. Enge, *Global Positioning System: Signals, Measurements and Performance Second Edition*.

[3] K. Chintalapudi, A. Padmanabha Iyer, and V. N. Padmanabhan, “Indoor localization without the pain,” in *Proceedings of the sixteenth annual international conference on Mobile computing and networking*. ACM, 2010, pp. 173–184.

[4] M. Youssef and A. Agrawala, “The horus wlan location determination system,” in *Proceedings of the 3rd international conference on Mobile systems, applications, and services*. ACM, 2005, pp. 205–218.

[5] A. LaMarca, Y. Chawathe, S. Consolvo, J. Hightower, I. Smith, J. Scott, T. Sohn, J. Howard, J. Hughes, F. Potter *et al.*, “Place lab: Device positioning using radio beacons in the wild,” in *International Conference on Pervasive Computing*. Springer, 2005, pp. 116–133.

[6] A. T. Mariakakis, S. Sen, J. Lee, and K.-H. Kim, “Sail: single access point-based indoor localization,” in *Proceedings of the 12th annual international conference on Mobile systems, applications, and services*. ACM, 2014, pp. 315–328.

[7] S. Beauregard and H. Haas, “Pedestrian dead reckoning: A basis for personal positioning,” in *Proceedings of the 3rd Workshop on Positioning, Navigation and Communication*, 2006, pp. 27–35.

[8] E. Costa-Montenegro, F. J. González-Castaño, D. Conde-Lagoa, A. B. Barragáns-Martínez, P. S. Rodríguez-Hernández, and F. Gil-Castiñeira, “Qr-maps: An efficient tool for indoor user location based on qr-codes and google maps,” in *2011 IEEE Consumer Communications and Networking Conference (CCNC)*. IEEE, 2011, pp. 928–932.

[9] J. J. Diaz, R. d. A. Maues, R. B. Soares, E. F. Nakamura, and C. M. Figueiredo, "Bluepass: An indoor bluetooth-based localization system for mobile applications," in *Computers and Communications (ISCC), 2010 IEEE Symposium on*. IEEE, 2010, pp. 778–783.

[10] K. Kaemarungsi and P. Krishnamurthy, "Modeling of indoor positioning systems based on location fingerprinting," in *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*, vol. 2. IEEE, 2004, pp. 1012–1022.

[11] L. M. Ni, Y. Liu, Y. C. Lau, and A. P. Patil, "Landmarc: Indoor location sensing using active rfid," *Wireless Networks*, vol. 10, no. 6, pp. 701–710, 2004. [Online]. Available: http://dx.doi.org/10.1023/B:WINE.0000044029.06344.dd

[12] J. A. Puertolas-Montañez, A. Mendoza-Rodriguez, and I. Sanz-Prieto, "Smart indoor positioning/location and navigation: A lightweight approach," *International Journal of Interactive Multimedia and Artificial Intelligence*, vol. 2, no. 2, pp. 43–50, 06/2013 2013. [Online]. Available: http://www.ijimai.org/journal/sites/default/files/files/2013/06/ijimai20132_2_5_pdf_27975.pdf

[13] R. Want, A. Hopper, V. Falcao, and J. Gibbons, "The active badge location system," *ACM Transactions on Information Systems (TOIS)*, vol. 10, no. 1, pp. 91–102, 1992.

[14] P. Fuhrer, D. Guinard, and O. Liechti, *RFID: from concepts to concrete implementation*. Department of Informatics-University of Fribourg, 2006.

[15] C. Medina, J. C. Segura, and Á. De la Torre, "Ultrasound indoor positioning system based on a low-power wireless sensor network providing sub-centimeter accuracy," *Sensors*, vol. 13, no. 3, p. 3501, 2013. [Online]. Available: http://www.mdpi.com/1424-8220/13/3/3501

[16] N. B. Priyantha, "The cricket indoor location system," Ph.D. dissertation, Cambridge, MA, USA, 2005, aAI0808861.

[17] I. C. S. L. M. S. Committee *et al.*, "Wireless lan medium access control (mac) and physical layer (phy) specifications," 1997.

[18] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani, "Energy consumption in mobile phones: a measurement study and implications for network applications," in *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*. ACM, 2009, pp. 280–293.

[19] I. Nabney, *NETLAB: algorithms for pattern recognition*. Springer Science & Business Media, 2002.

A

Cycle-to-Shop

Here, we present the Cycle-to-Shop's Mockups to illustrate how we idealized the application and some print screens to show how it ended.

A.1 Mockups

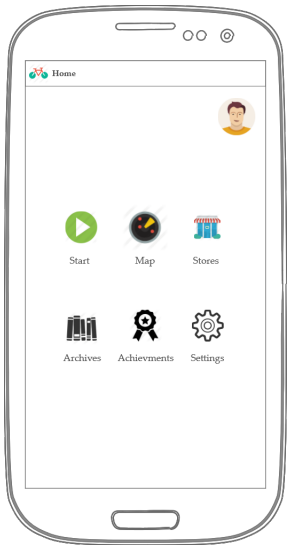

Figure A.1: Main menu.

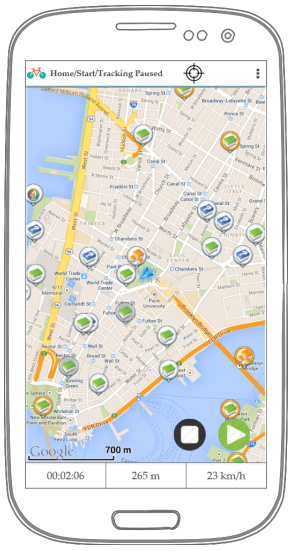

Figure A.2: Map.

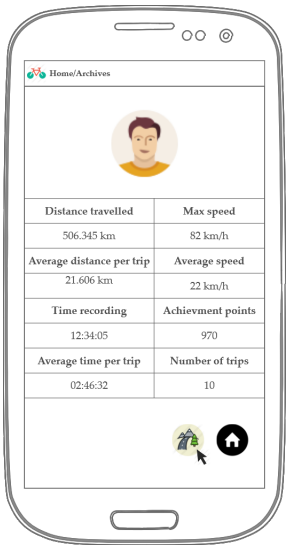

Figure A.3: User profile.

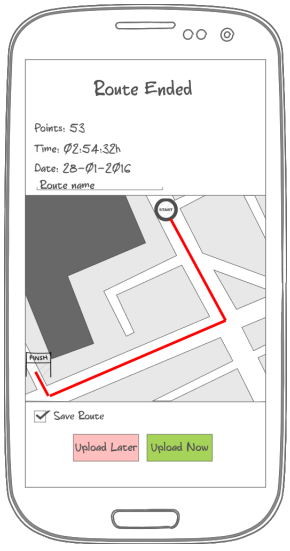

Figure A.4: Route resume.

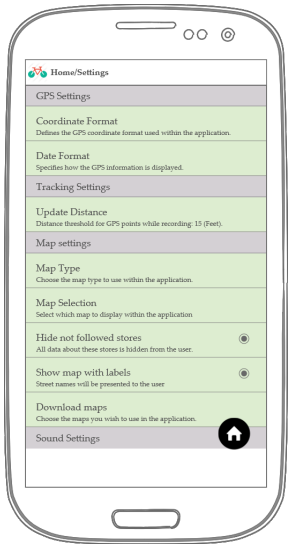

Figure A.5: Settings.

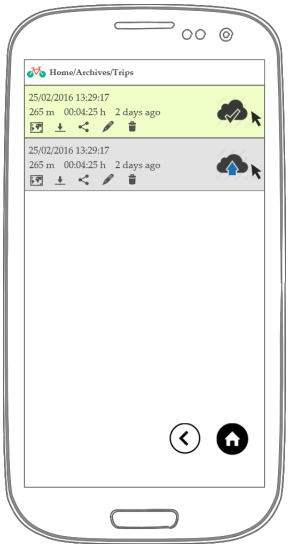

Figure A.6: Routes list.

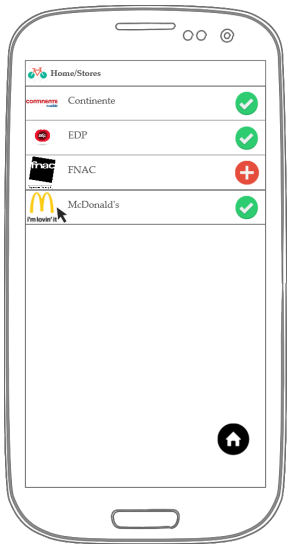

Figure A.7: Stores, i.e. partners.

Figure A.8: Stores' challenges.

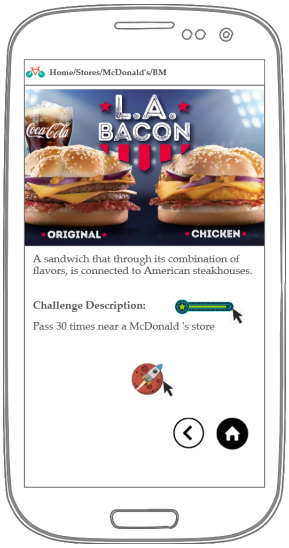

Figure A.9: Challenges' description.

A.2 Print screens

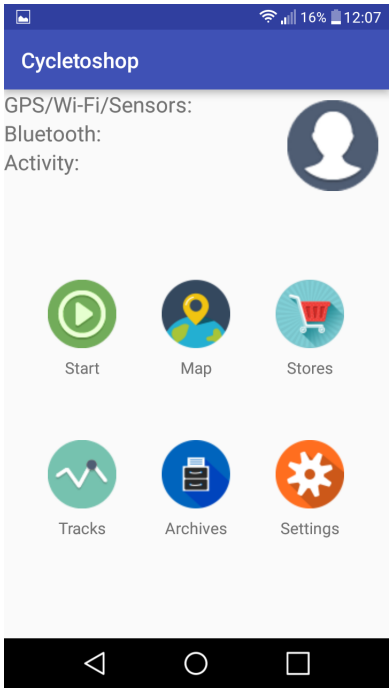

Figure A.10: Figure A.1 implementation.

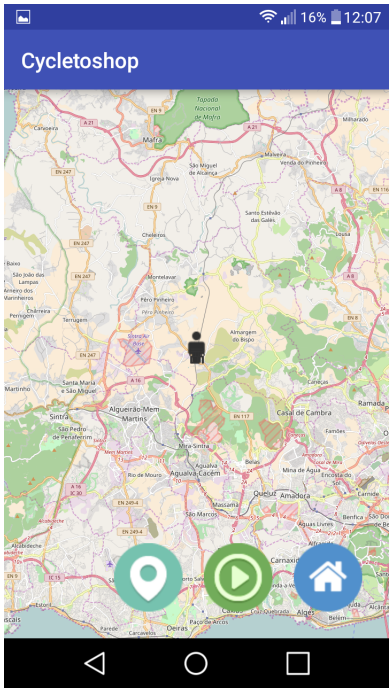

Figure A.11: Figure A.2 implementation.

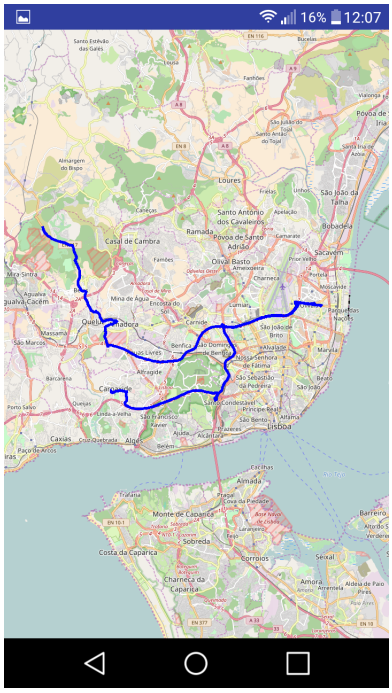

Figure A.12: Figure A.4 implementation.

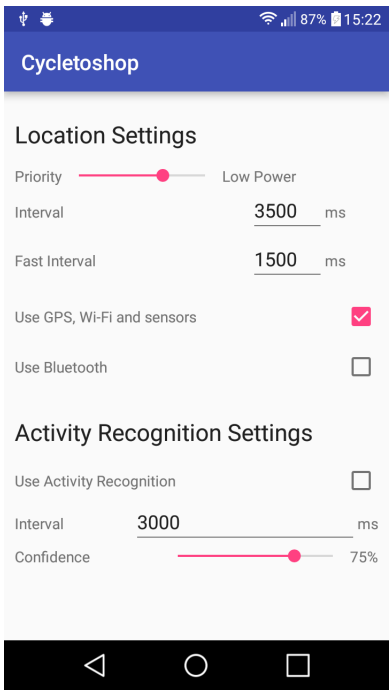

Figure A.13: Figure A.5 implementation.

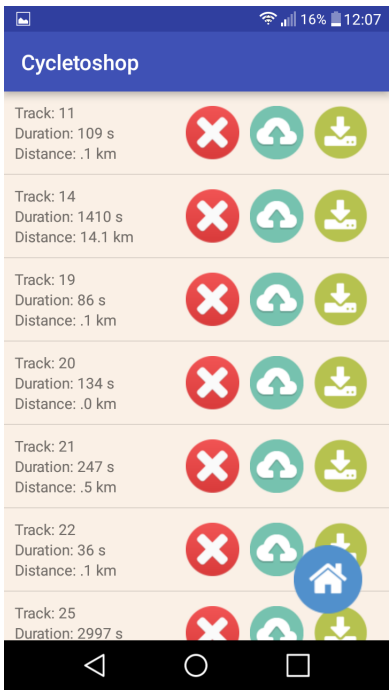

Figure A.14: Figure A.6 implementation.

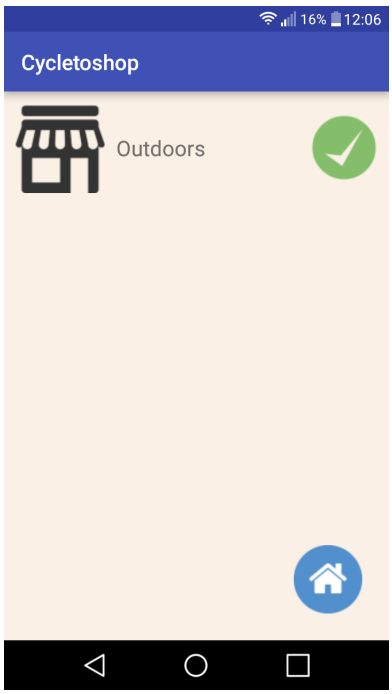

Figure A.15: Figure A.7 implementation.