MultiTrack: Flexible Use of Multiple Location Technologies

Rui Santos, João Barreto, Paulo Ferreira
rddsantos@gmail.com, [joao.barreto,pjpf]@tecnico.ulisboa.pt

INESC-ID, Instituto Superior Técnico, Universidade de Lisboa

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 providing location information for these services, but they all focus on particular scenarios by applying specific location technologies and/or techniques which they consider more suited for those cases. 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. 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 Cycle-to-Shop's requirements and its goal of supporting the flexible usage of different location technologies and/or techniques by LBSs.

Keywords: Location-based Services, positioning systems, technologies, techniques, flexible, MultiTrack, Cycle-to-Shop.

1 Introduction

As wireless networks and mobile devices evolve, users demand more sophisticated applications. LBSs 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. However, increasing the accuracy of this information has costs, usually in terms of energy and infrastructure.

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 [1–11,13,14] capable of providing location information for LBSs. These approaches focus on specific scenarios, e.g. indoors or outdoors, by applying specific location technologies and/or techniques that allow them to obtain the location information with the highest accuracy and lowest energy and infrastructure costs for those cases. However, if we use these systems at different scenarios, their results are highly affected, i.e. either the accuracy of the collected location information degrades or the infrastructure costs grow, e.g. the Global Positioning System (GPS), which is the standard positioning system for outdoors but for indoors it is not recommended, since its accuracy is highly degraded by physical structures, which are frequent in those scenarios.

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.

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 and that allows LBSs to control how and when these approaches are used. When LBSs interact with the framework, they choose the location technologies and/or techniques applied and the corresponding configurations, e.g. the rate at which a location technology and/or technique acquires information.

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: i) Modularity, i.e. the modules that implement the location technologies and/or techniques supported need to be easily added and removed, ii) 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, and ii) Energy Efficiency, i.e. the energy spent by the framework need to be minimal since it runs on mobile devices, which are battery constrained.

To test if MultiTrack achieves its goal, we developed a LBS called Cycle-to-Shop. The Cycle-to-Shop rewarding application is an online LBS that runs in smartphones and is associated with the TRACE project[1]. It requires location information to motivate users to cycle more frequently by rewarding them based on the distance they do and on the places they have been, which are usually stores from partners that support the application. It relies on 3 basic assumptions: i) users carry mobile smartphones with location capabilities, e.g GPS, Wi-Fi and

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

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 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. It also requires the correct detection of users' modality, since it can only reward users that are cycling, and minimal energy consumption and infrastructure costs, to ease its development and acceptance.

The remainder of the document is organized as follows. In Section 2, we describe other approaches capable of providing location information for LBSs, enhancing their results at the light of our requirements. The idea is to understand their differences to determine which location technologies and/or techniques should be first supported in MultiTrack. In Section 3, we describe the MultiTrack architecture, focusing on its main components and interactions and enhancing how they contribute for supporting the flexible usage of different location technologies and/or techniques. In Section 4, we describe the implementation choices made and in Section 5 the experimental results obtained. Finally in Section 6, we present the conclusions and MultiTrack's limitations and future work.

2 Related Work

There are many different approaches when it comes to localization, for example, Wi-Fi based positioning systems, e.g. RADAR [1] and Horus [14], which take advantage of the wide Wi-Fi network already deployed for other services like Internet and multimedia. These systems are able to acquire accurate location information with low energy and infrastructure costs. However, they are mainly used for indoors because it is where Wi-Fi networks are usually available.

Positioning systems based on Infrared [13] (IR), Ultrasound [9,12], Radio Frequency Identification [6] (RFID), Bluetooth[2] and Quick-Response [4] (QR) codes are examples of schemes that are usually more accurate than Wi-Fi but that require specialized infrastructure, which raises the infrastructure costs required to deploy these approaches. For this reason, they are mainly used indoors, where the covered areas are smaller.

On the other hand there is the GPS [10], which is currently the standard positioning system for outdoors mainly due to its high accuracy, availability and low infrastructure costs for users. However, for indoors GPS is not a viable approach because its signals are highly affected by physical structures, which degrade the accuracy of the collected location information. Nevertheless, the main issue related with GPS is still the energy consumption, which is a critical factor for mobile LBSs.

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

Finally, there are the Hybrid positioning systems, e.g. EZ [3], SAIL [8], Pedestrian DR [2] and UnLoc, which similarly to MultiTrack, apply different location technologies and/or techniques to collect location information. On Table 1 we present an overview of these approaches, enhancing their results in terms of accuracy, energy consumption and infrastructure costs. The idea is to understand their differences to determine which should be first supported in MultiTrack. Moreover, 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 1: Positioning systems overview.

As we can see, 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.

Considering that the Cycle-to-Shop rewarding application 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 first in MultiTrack. Having this into account, we opted by supporting GPS, Wi-Fi, mobile sensors (accelerometer, gyroscope and magnetometer) and Bluetooth because together, they are able to provide location information with an accu-

racy on average better than 10 m, both indoors and outdoors and with minimal energy and infrastructure costs, thus allowing us to achieve Cycle-to-Shop's requirements. Nevertheless, this does not mean that the remaining location technologies and/or techniques should be discarded from MultiTrack. As we have seen, all approaches have different characteristics, i.e. advantages and disadvantages, and so, all are valuable solutions from providing location information for LBSs.

3 Architecture

The MultiTrack architecture (see Figure 1) can be broken into the following components: i) the Tracker, ii) the Persistent Track Storage, and iii) the Client (singleton). Next, we describe these components and their interactions (including those with external entities, e.g. the Mobile Hardware), enhancing how they achieve MultiTrack's architectural design requirements.

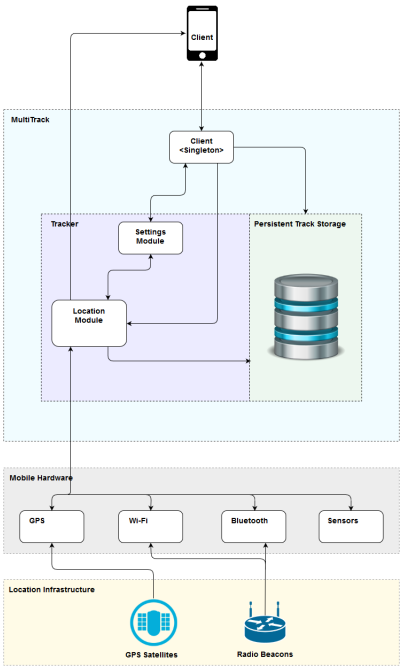

Fig. 1: The MultiTrack architecture.

3.1 Tracker

The Tracker is responsible for collecting location information from available sources, e.g. GPS, Wi-Fi, Bluetooth, QR-codes 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.

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 1.1), that is implemented by each of its sub-modules.

Listing 1.1: Interface provided by the Location Module.

```
public interface LocationInterface {
    void start();
    void stop();
}
```

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.

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.

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 for 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 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.

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.

4 Implementation

There are several platforms powering mobile smartphones today, e.g. Android, iOS, Windows 10 mobile and Blackberry. According to IDC[3], 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.

4.1 Location Technologies and APIs

As already said on Section 2, the location technologies and/or techniques that must be initially supported in MultiTrack are the GPS, Wi-Fi, Bluetooth and mobile sensors (accelerometer, gyroscope and magnetometer). To implement GPS and Wi-Fi the following APIs arise: i) android.location[4], and ii) Google Location Services[5].

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.

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.

[3] http://www.idc.com/prodserv/smartphone-os-market-share.jsp
[4] https://developer.android.com/reference/android/location/package-summary.html
[5] https://developers.google.com/android/reference/com/google/android/gms/location/package-summary

Regarding the usage of Bluetooth Beacons in Android, we opted by using the Estimote Beacons[6] because they have a simpler and more intuitive API than the other approaches, e.g. Blueup[7] and Kontakt[8].

4.2 Client's Configurations

The Settings Module stores Location's sub-modules configurations and allows clients to control them (see Figures 2 and 3), 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. The Interval sets the time in milliseconds between location updates. The Fastest Interval sets a limit for the Interval parameter, since several LBS might be requesting location updates at different rates. The Priority establishes the focus of the location request, i.e the balance between accuracy and 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. 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.

4.3 Track Storage

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

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

5 Experimental Results

Several experiments were made to determine if MultiTrack achieves the requirements presented in Section 1 and consequently, the goal of supporting the flexible usage of different location technologies and/or techniques by mobile smartphone LBSs. In this section we present and describe those experiments, enhancing their results at the light of our requirements.

5.1 Experiments

The following experiments were done to test the framework: 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 2). The Low Power configuration only changes the priority parameter to "Low Power" relative to the High Accuracy configuration (see Figure 3).

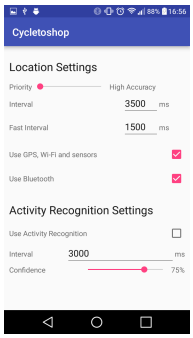

Fig. 2: Cycle-to-Shop configuring MultiTrack for High Accuracy.

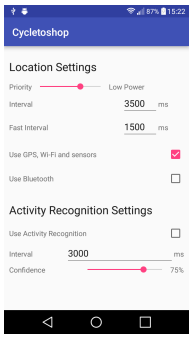

Fig. 3: Cycle-to-Shop configuring MultiTrack for Low Power consumption.

The Accuracy Experiment has 2 objectives: i) test the accuracy of the location information collected by MultiTrack to determine if the framework achieves Cycle-to-Shop's accuracy requirements, and ii) test if MultiTrack achieves the Technologies Control requirement presented on Section 1. The experiment consists on a user walking in circles 3 m away from store's coordinates, both indoors

and outdoors 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.

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.

Results obtained indoors (see Figure 4) 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.

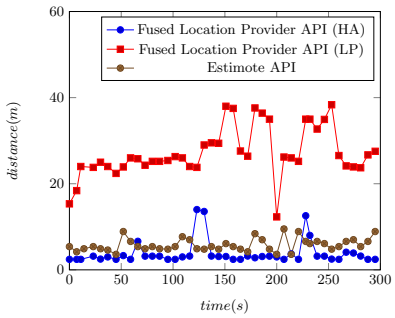

Fig. 4: Distance in meters between the user and store's coordinates acquired by MultiTrack indoors using either the Fused Location Provider API, configured both for High Accuracy (HA) and Low Power (LP), and the Estimote API.

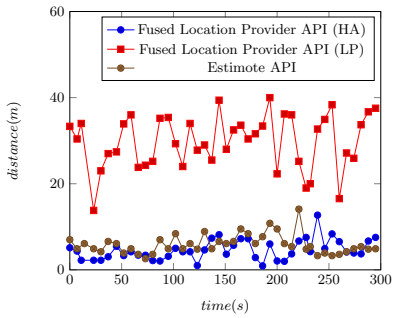

Fig. 5: Distance in meters between the user and store's coordinates acquired by MultiTrack outdoor using either the Fused Location Provider API, configured both for High Accuracy (HA) and Low Power (LP), and the Estimote API.

Results obtained outdoors (see Figure 5) 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 Costs Experiment has the objective of determining the energy and infrastructure costs required by MultiTrack to collect location information, to test if it achieves the Energy Efficiency requirement presented on Section 1. It consists on a user driving a car through an urban outdoors route with 14.2 km distance according to Google Maps[9], 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.

Results obtained (see Figure 6) 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 %.

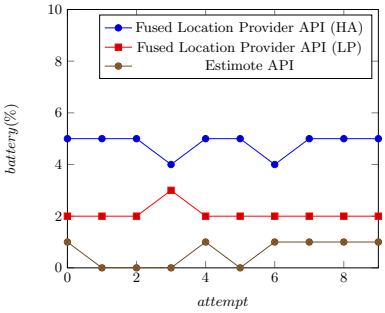

Fig. 6: Percentage of device's battery spent by the application using MultiTrack.

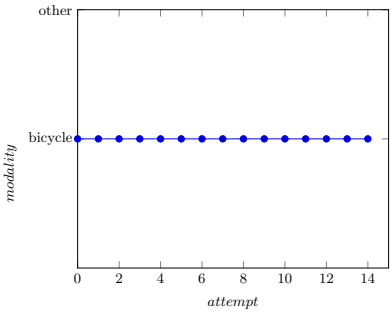

Fig. 7: User's modality collected by MultiTrack using the Activity Recognition API.

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

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).

The User's Activity Detection Experiment has the objective of testing 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.

Results obtained during the User's Activity Experiment (see Figure 7) show us that MultiTrack successfully detected when the user passed nearby the store while cycling.

6 Conclusions

Experimental results show 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 Section 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 Section 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.

At last, we can conclude that MultiTrack successfully achieves Cycle-to-Shop's requirements since: i) on average it provides location information with an accuracy less than 10 m, both indoors and outdoors, ii) it successfully detects when users are cycling, iii) it minimizes energy costs by supporting the flexible usage of energy efficient location APIs, 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.

6.1 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 LBS, 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 LBS requirements, we cannot simply support the flexible usage of GPS, Wi-Fi, Bluetooth and mobile sensors, because as we have seen on Chapter **??**, the remaining location technologies and/or techniques collect location information with different characteristics, which can be useful for LBS. 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 LBS requirements and environmental characteristics. At the moment, LBS 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 LBS expectations.

References

1. Paramvir Bahl and Venkata N. Padmanabhan. Radar: An in-building rf-based user location and tracking system. In *INFOCOM*, pages 775–784, 2000.
2. Stephane Beauregard and Harald Haas. Pedestrian dead reckoning: A basis for personal positioning. In *Proceedings of the 3rd Workshop on Positioning, Navigation and Communication*, pages 27–35, 2006.
3. Krishna Chintalapudi, Anand Padmanabha Iyer, and Venkata N Padmanabhan. Indoor localization without the pain. In *Proceedings of the sixteenth annual international conference on Mobile computing and networking*, pages 173–184. ACM, 2010.
4. Enrique Costa-Montenegro, Francisco J González-Castaño, David Conde-Lagoa, Ana Belén Barragáns-Martínez, Pedro S Rodríguez-Hernández, and Felipe 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)*, pages 928–932. IEEE, 2011.
5. Javier JM Diaz, Rodrigo de A Maues, Rodrigo B Soares, Eduardo F Nakamura, and Carlos MS Figueiredo. Bluepass: An indoor bluetooth-based localization system

for mobile applications. In *Computers and Communications (ISCC), 2010 IEEE Symposium on*, pages 778–783. IEEE, 2010.
6. Patrik Fuhrer, Dominique Guinard, and Olivier Liechti. *RFID: from concepts to concrete implementation*. Department of Informatics-University of Fribourg, 2006.
7. Anthony LaMarca, Yatin Chawathe, Sunny Consolvo, Jeffrey Hightower, Ian Smith, James Scott, Timothy Sohn, James Howard, Jeff Hughes, Fred Potter, et al. Place lab: Device positioning using radio beacons in the wild. In *International Conference on Pervasive Computing*, pages 116–133. Springer, 2005.
8. Alex T Mariakakis, Souvik Sen, Jeongkeun Lee, and Kyu-Han Kim. Sail: single access point-based indoor localization. In *Proceedings of the 12th annual international conference on Mobile systems, applications, and services*, pages 315–328. ACM, 2014.
9. Carlos Medina, José Carlos Segura, and Ángel De la Torre. Ultrasound indoor positioning system based on a low-power wireless sensor network providing sub-centimeter accuracy. *Sensors*, 13(3):3501, 2013.
10. Pratap Misra and Per Enge. *Global Positioning System: Signals, Measurements and Performance Second Edition*.
11. Lionel M. Ni, Yunhao Liu, Yiu Cho Lau, and Abhishek P. Patil. Landmarc: Indoor location sensing using active rfid. *Wireless Networks*, 10(6):701–710, 2004.
12. Nissanka Bodhi Priyantha. *The Cricket Indoor Location System*. PhD thesis, Cambridge, MA, USA, 2005. AAI0808861.
13. Roy Want, Andy Hopper, Veronica Falcao, and Jonathan Gibbons. The active badge location system. *ACM Transactions on Information Systems (TOIS)*, 10(1):91–102, 1992.
14. Moustafa Youssef and Ashok Agrawala. The horus wlan location determination system. In *Proceedings of the 3rd international conference on Mobile systems, applications, and services*, pages 205–218. ACM, 2005.