Energy-Efficient Location-Based System Architecture for Mobile Client-Server Applications using Location-Dependent Content

Maximilian J. Landsmann

MASTERARBEIT

eingereicht am
Fachhochschul-Masterstudiengang

Interactive Media

in Hagenberg

im September 2013

Declaration

I hereby declare and confirm that this thesis is entirely the result of my own original work. Where other sources of information have been used, they have been indicated as such and properly acknowledged. I further declare that this or similar work has not been submitted for credit elsewhere.

Hagenberg, September 30, 2013

Maximilian J. Landsmann

Contents

Abstract

With the increasing computational power and positioning capabilities of smartphones, the number of location-based applications grew as well. Not only do many of them now present location-based data comprehensively on digital maps, but they also track the user's location continuously in the background for their respective purposes. As the use of location sensors—especially the GPS receiver—in mobile phones consumes a high amount of energy, this thesis proposes an architecture that makes use of user-relevant and location-dependent data to recognise relevant locations and dynamically adjust the accuracy of the positioning systems accordingly. With the use of the results of a series of preliminary tests concerning energy consumption, accuracy of location sensors and network connections on mobile phones, the architecture is designed to maximise energy-efficiency while preserving accuracy at user-relevant locations. The application *Letterbox*—a location-based messenger—is presented in this thesis and its use of the architecture shows an increase in energy-efficiency in most scenarios while still offering accurate positioning at the relevant locations at limited velocities. The architecture therefore has high potential value for various mobile applications using location-dependent content.

Kurzfassung

Durch die fortwährend steigende Rechenleistung und Möglichkeit der Positionierung auf Smartphones, vermehren sich zunehmend ortsbasierte Anwendungen. Diese präsentieren Daten auf digitalen Karten und bestimmen die Position des Benutzers kontinuierlich im Hintergrund für ihre jeweiligen Zwecke. Da jedoch die Positionierungssysteme in mobilen Geräten einen hohen Energieverbrauch aufweisen, wird in dieser Masterarbeit eine Architektur vorgestellt, welche für den Benutzer relevante und ortsabhängige Daten verarbeitet. Aufgrund dieser wird die Genauigkeit der Positionierungssysteme dynamisch angepasst. Die Architektur wird basierend einer Reihe von Tests bezüglich Energieverbrauch und Genauigkeit der involvierten Sensoren entworfen. Dies soll zum Ziel führen, dass Energieeffizienz maximiert, aber trotzdem eine hohe Genauigkeit für die Abfrage von ortsbasierten Daten ermöglicht wird. *Letterbox* – eine Applikation für das Senden von ortsbasierten Nachrichten – wird in dieser Arbeit vorgestellt. Es wird gezeigt, in welchem Ausmaß die entworfene Architektur eine Erhöhung der Energieeffizienz an den für den Benutzer relevanten Positionen verursacht. Die Architektur zeigt einen hohen potentiellen Wert für unterschiedliche mobile ortsbasierte Applikationen.

Chapter 1

Introduction

Mobile phones have advanced greatly in recent years. Progressively, the screens turned to touch-screens, pixels decreased in size and with it computational power increased exponentially. Considering the location capabilities and the fact that smartphones change from being the exception to being dominant in the mobile phone sector,[1] location-based applications increasingly gain significance.

1.1 Location-Based

Since the introduction of location sensors to mobile phones, one of the most common features in applications around the globe was to use location-based content. Examples for such applications are mostly found in gastronomy, shops, tourism or any other information about services at fixed places. The combination of these with map services allow for very comprehensible presentation of the data and it puts the information into context. An example for this is the app *Yelp*,[2] presenting information about bars and restaurants (Figure 1.1).

1.2 User-Based

The next step of these applications was to include the user's own location and thus adding a substantial amount of context to them. As the user nowadays usually has her phone with her at all times, it has become quite significant for applications to show the user-relevant location-based data in her vicinity. This can also be seen with the app *Yelp* in Figure 1.1.

One step even further, as some applications have gone, is to make the application continuously aware of its location. An example for this would

[1] http://www.nielsen.com/us/en/newswire/2013/mobile-majority--u-s--smartphone-ownership-tops-60-.html

[2] http://www.yelp.co.uk/

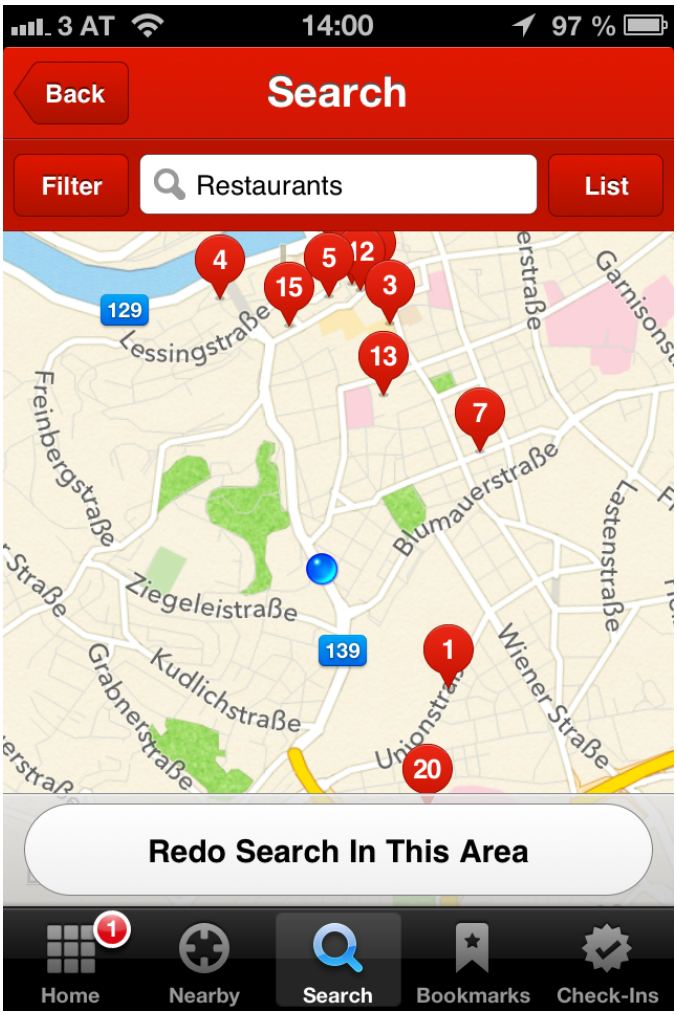


Figure 1.1: Location-based data presented on a map in *Yelp*, a gastronomy app.

be the lifestyle and social gaming app *Foursquare*.[3] It not only shows places on a map around the user, but also makes continuous location updates in the background and notifies the user, should she be near one of his favourite places. This suggestive behaviour of the app is applicable in numerous other scenarios to give the user information specific to her current location. Examples for this would include location-based reminders, suggestions of interesting information for the user or any other kind of location-based notifications. However, as this is applied foremost in mobile devices, some problems arise with this thought of continuous location awareness.

[3] https://foursquare.com/

1.3 High Energy Consumption

A limitation which every mobile electronic device has is its battery. It has a limited capacity, making power management a substantial element of mobile application development. This starts at the development of the operating system as well as with mindful implementation of the programs.

When it comes to location-based applications, this is a particularly significant aspect, because location sensors—especially GPS receivers— consume a high amount of energy and when used continuously are known to drain the battery of a mobile device in only a few hours. It is unacceptable that one application—especially when running in the background and sometimes without the users knowledge—uses up all the battery, rendering the device practically useless and all but removes its mobile aspect.

This problem is so fundamental that the manufacturers and developers of mobile operating systems quickly introduced cheaper ways of localisation in terms of energy. The most common of these being via WiFi and cell towers. They allow quicker and cheaper localisation, however at the cost of accuracy. The problem of accurate continuous location updates therefore still remains.

1.4 Proposed Solution

When considering the problem of energy consumption of the previous section and focusing on suggestive user-based applications as described in Section 1.2, one simple question arises: does the application need accurate location information at all times?

In the case of user-based applications it is often the case that only user-relevant data is presented. When only location-based applications are considered, this user-relevant data would then also be location-dependent. This means that there is a set of relevant locations as opposed to all other regions not containing such locations.

As an example, the application *Letterbox* was developed during the course of this master thesis. It is essentially a location-based messaging service. A message is not only sent to a specific person, but also to a specific place, making the app both user- and location-based. When a user goes to a place where a message has been left for her, the application informs her about it. It is therefore necessary for the application to always track the users location, however only the places where messages have been put are relevant and exact location information is only needed in the vicinity of such places.

The proposed solution to this problem is a continuous adjustment of accuracy of the location sensors to save energy at locations where no relevant data is present and still make use of exact measurements when precise positioning is necessary.

1.5 Method

To effectively implement this solution and to be able to evaluate its practical value, a number of steps are taken. A way to accurately measure the energy consumption is needed in order to find a compromise between accuracy and energy-efficiency. A series of preliminary tests are conducted to establish how much energy each of the technologies and sensors involved consume under variant circumstances. These are foremost location services like GPS, WiFi and cellular, however tests also need to be made when the phone is running idle, to provide a baseline for the results.

The proposed architecture is then implemented in the *Letterbox* project and again similar energy tests are conducted. The results are expected to show that dynamically changing technologies according to relevant data will minimise energy cost without sacrificing accuracy at significant locations.

1.6 Structure

This thesis is divided into five main chapters, apart from the introduction. First, the background chapter describes the current location sensors in modern mobile phones and presents related work in the field. The chapter of preliminary tests then describes the tests that are made to determine the energy usage and performance of the positioning systems and presents their results. The following chapter gives an overview of the project *Letterbox* and how the proposed architecture is implemented, taking the findings of the preliminary tests into account. The experiment chapter defines the conditions under which the architecture is tested and presents the results, which are then discussed in the evaluation chapter.

Chapter 2

Background

2.1 Location

Before trying to build a system to minimise energy consumption for applications using location-based user-content, it is important to be aware of how location sensors in mobile devices operate. By considering the different technologies, their accuracies and energy consumption, the advantages and drawbacks of each can be used to try to achieve the goal of energy-efficiency.

2.1.1 Cellular

Although it might not seem obvious at first, location management in mobile phones has been a central aspect since the introduction of the mobile phone. When it receives a call, the network of cell towers must know where to broadcast the signal in order for the call to find its target. Therefore the network has to have certain knowledge of the phone's whereabouts and from there it was a small step to the phone being aware of its approximate location as well. A detailed explanation of the process as well as several modern approaches to track the phones location with respect to the cellular network can be found in [5].

However, cellular localisation nowadays goes much further than that. As the coverage and density of cell towers grew, a mobile phone may be in range of multiple cell towers at once. This allows for cell triangulation. The distance to each tower can be approximated and due to the fact that the locations of the towers are known, the location of the phone and its user can be calculated as well. A more detailed description of this method is given in [2].

2.1.2 GPS

The Global Positioning System, commonly known as GPS, has become the most popular, most accurate global positioning system since it became fully

operational and publicly available for manufacturers and developers to integrate it into their devices and applications.

The system consists of numerous satellites, constantly orbiting the earth and sending signals towards it. These signals contain information about time and location of the satellite. GPS receivers can interpret these signals and by measuring the time the signal needed to reach the GPS receiver it can calculate the distance to the corresponding satellite. This is done with multiple signals simultaneously, allowing the receiver to calculate its own position by triangulation.

The GPS is not the only global positioning system in existence. As it is managed by the United States government, other nations felt the need to create their own systems. Therefore the Russian system *GLONASS*, the European *Galileo* and the Chinese *Compass* were created or at least are planned to go fully operational within the coming decade. For the sake of simplicity, the term GPS will be used to describe any active global positioning system using satellites throughout the rest of this thesis because it is still the most prominent in smartphones.

2.1.3 WiFi

With the rapid spread of WiFi—a standardised WLAN (Wireless Local Area Network) technology—to the point where in urban areas it is not unusual to be continuously in reach of multiple WiFi hotspots when walking down a street, WiFi localisation gained in significance.

Chen and Kobayashi argue in a similar matter in [3] when proposing a model for using the hotspot's signal strength for geolocation, which is now common for WiFi localisation in mobile devices. By approximating the distance to each hotspot which the phone is in range of, it can again use triangulation to calculate its position. This technology is most effective in urban areas, due to the higher density of WiFi networks.

For this to work, it is essential that the location of these hotspots is known. To include as many WiFi networks as possible, companies like *Apple* and *Google* have implemented in the operating systems of their mobile products a way to track each phone's connections to WiFi hotspots and the current location, quickly creating a crowd-sourced database of WiFi hotspot locations. *Apple* confirms and explains this in a press release in 2011.[1] The most important advantages of the WiFi positioning system are its occasional superiority in urban areas and that it is known to need less energy than a GPS receiver.

[1] http://www.apple.com/pr/library/2011/04/27Apple-Q-A-on-Location-Data.html

2.2 Accuracy and Coverage

The positioning technologies described in the previous section all have their advantages and disadvantages. Two of the most important ones are related to accuracy and coverage.

Concerning coverage, the clear winner is GPS, as it is accessible in populated areas as well as in the middle of the pacific ocean. Next in line is cellular positioning, as it is spread widely into rural areas. WiFi localisation only works with WiFi hotspots nearby, usually limiting it to inhabited areas. Nevertheless, all of them share the same weakness. The device needs to be able to receive the signals from satellite, cell tower or hotspot, which means that often there are holes in the coverage of each system, usually indoors or underground. WiFi here can sometimes have an advantage, often being available indoors as well.

The accuracy of the technologies is another matter. Using cell towers, positioning can be quite vague, if only one of them is nearby. However, it becomes more and more accurate as more cell towers are in reach of the phone's sensors.

The same way cellular positioning accuracy increases with the number of towers, WiFi localisation accuracy increases with the number of hotspots as well. This is the case in any triangulation, as errors can be minimised with increasing number of measurements. In [3], Chen and Kobayashi obtain results in their signal strength based approach shown in Table 2.1.

Nr. of Hotspots	Max. Error (meters)	Min. Error (meters)
2	15.98	2.75
3	2.64	2.12
4	2.80	2.06
5	2.80	1.41

Table 2.1: Results of location errors in Chen's and Kobayashi's [3].

Clearly visible in these results is the increase in accuracy when increasing the available hotspots. However, it is important to realise, that the actual values of the error cannot be expected to apply to real world scenarios. The tests were made in a controlled environment and Chen and Kobayashi themselves come to the conclusion that "the number and proper placement of APs" (access points) is a factor controlling the location error. This cannot be controlled in common every-day user applications, so the accuracy will mostly be substantially lower.

With GPS, it is commonly known that it can provide accuracy well within 10 meters and as described in [1], GPS calculates the position through time-stamp comparison of satellites and measurements of how long the signal

travels to the receiver. The accuracy therefore depends highly upon the satellites' clocks. They describe the complex process of keeping the time on the satellites like this:

> *A Kalman filter software program estimates the time error, frequency error, frequency drift and Keplerian orbit parameters for each of the satellites and its operating clock. This information is uploaded to each satellite so that it can be broadcasted in real time. This process provides GPS time consistency across the constellation to within a small number of nanoseconds and accurate position determination of the satellites to within a few meters.*[1]

Furthermore, GPS is known to have greater errors in urban areas, where high buildings are present. As GPS receivers need a clear line of sight to the satellites, high buildings block signals from those positioned sideways to the receiving device therefore reducing the number of satellites available for the calculations. This problem is the basis of the work of Paek, Kim and Govindan in [9].

2.3 Energy Consumption

With the ever-growing market of mobile devices the significance of energy cost and management is rising proportionally as well. When considering the proposed solution in Section 1.4, a number of factors must be considered with regard to energy-efficiency in mobile devices.

Firstly and most importantly, the location sensors usually draw the most power. Foremost the GPS sensor can reduce the device's battery life significantly. This is not surprising when considering that the mobile device needs to receive signals from multiple satellites simultaneously over great distances and perform a high amount of calculations. This is one of the reasons additional positioning technologies like cellular and WiFi localisation have been developed. However it is essential to the proposed solution to be aware of all their energy characteristics. WiFi and cellular positioning are considered more energy efficient, as confirmed by numerous works in the field including [4] and [7]. In the case of cellular localisation this seems to make perfect sense, seeing that mobile phones are connected to the cell network anyway and it does not require the use of additional sensors.

However, also other factors must be taken into consideration. One of them being Internet connection. The heavy use of server requests will have an effect on energy consumption, the extent of which depending on the connection type currently active—be it WiFi, Edge or 3G—and the interval at which requests are sent.

2.4 Related Work

The limitations concerning location sensors in smartphones have lead to numerous works trying to find solutions to some of the problems involved. This definitely also includes the developers of the operating systems on mobile devices themselves, thriving to maximise efficiency on their platforms.

Some researchers and developers, however, go even further and continue to come up with different ways to improve performance in the location sector. An interesting survey on the matter focuses on energy-efficient location sensing systems and generally distinguishes from *dynamic tracking* solutions which "attempt to minimise the frequency of needed position updates by only sampling positions (generally with GPS) when the estimated uncertainty in position exceeds the accuracy threshold" [8] as opposed to just using the same location sensors on the mobile device continuously. Researchers and developers have tried to find different scenarios where dynamic tracking can be applied to improve performance even further.

To use their terminology, the proposed solution in Section 1.4 would also fall in line with the *dynamic tracking* technologies, adding the aspect of user-relevant location-based data.

2.4.1 Related Problems

The most pressing problem is that of energy consumption. As mentioned in the previous sections and also described in [4] and [10], WiFi and cellular localisation need considerably less energy, however are much less accurate.

This leads to the next issue, which is accuracy. Location accuracy for one depends on the technology used, but even more important are the current conditions the phone or mobile device is situated in. The main issue therefore is actually the unpredictable accuracy.

Another less urgent problem is the time it takes to get a location fix. The different technologies also vary in this aspect, making it another possible parameter in the *dynamic tracking* technologies.

2.4.2 Existing Solutions

The related problems have led to numerous researchers trying to find solutions for different scenarios. This thesis is not the first work recognising the significance of the energy-accuracy trade-off.

Dynamic Tracking and Prediction

In [4], the authors present the energy-efficient localisation framework *EnLoc*. They argue for its significance with the same problems of location sensors in mobile phones as described in Section 2.4.1. They come to the conclusion that the use of GPS should be minimised to save energy and to make up

for loss of accuracy, they have a quite sociological approach. In detail, they aim to "Exploit habitual activity of individuals and behavior of populations to predict location" and "Evaluate heuristics in real life situations", meaning that by tracking the user for some time, a pattern of probable locations at specific times of the day can be mapped out. Through prediction algorithms the most probable location of the user is then calculated.

This approach is very promising, combining technological and habitual activity. For some applications who want to know mainly if the user is at one of her usual places this can drop energy usage significantly. The drawback, however, is that this system is based on prediction and probability. It may therefore predict your position wrong for quite some time before it realises its mistake, should one deviate from one's usual path.

Similar to this, *EnTracked* [6] also is a system based on prediction. However, they only consider pedestrian movements and it works rather locally. Furthermore, the prediction relies on the accelerometer and trajectory calculations, rather than habitual user behaviour.

a-Loc

The energy-accuracy trade-off is a fundamental part of most works in the field, however it is most prominent in the system *a-Loc*. It is a "system that can automatically tune the location energy and accuracy trade-off by continually adapting to the dynamic location sensor characteristics and application needs", as Lin describes it in [7]. The author presents extensive measurements on energy cost and accuracy of the different sensors on the mobile phone to strengthen his implementation decisions. His algorithm, simply put, tries "to determine the most energy efficient sensor to be used, such that the required location accuracy can be achieved." This principle is actually quite similar to the result this thesis is trying to achieve, except that the proposed architecture of this thesis focuses on user-relevant locations, revealing further possibilities for energy saving.

2.4.3 Practical Value

All of these systems and algorithms have a target application and are useful in their respective area. It might however be suitable to consider and compare their practical value. For instance, it is probable that *EnLoc* has a high amount of advantages concerning energy saving, but it is not as much of a position tracking system as it is a location prediction algorithm. When actual positions are needed—which is the case in the application *Letterbox*—it is not really applicable. Therefore it is of high relevance only in a smaller target area.

The system *a-Loc* is more widely useful, as it relies on variable accuracy. It can be configured to be more accurate or more energy efficient, while still

always relying on positioning technologies, rather than prediction. However, when high accuracy is set, the energy consumption also remains high, which is not an easy problem to solve.

Chapter 3

Preliminary Tests

To be able to more properly judge the effectiveness of the architecture, some preliminary tests are made to measure the energy consumption of the different components of the architecture proposed in this thesis. These are server communication, location services and, as a baseline, the phone running in idle state.

3.1 Energy Measurement

To measure the energy usage on the phone, the tool *Powergremlin*[1] is used. As described on their blog, "Powergremlin leverages private APIs to measure the battery's current capacity over time and determine current draw." This tool was chosen, because its use of private APIs allows quite low-level access to the data. Though the performance tool *Instruments*[2] certainly has numerous advantages when it comes to the development of applications, the use of private APIs promises a more reliable source of the data without as many abstractions and generalisations. The part of the private API used is the *GAIA* framework and is included in the method as shown in the code example 3.1.

The test program using this code has three views, one for each of the different tests. Each of these has multiple parameters that can be adapted to bring variations into each separate run. These views are shown in Figure 3.1. The parameters in the user interface allow for easy control over each test run without the need for a computer in between. The test phone used for all tests, if not explicitly stated otherwise, was an iPhone 4, three years old. An older model was selected due to availability and the lower battery capacity in hope for a more visible effect on the results when testing energy consumption. However, *iOS 6.1.3* was used at the time of development, to still make use of the latest technology as much as possible.

[1] http://blog.palominolabs.com/2012/11/14/introducing-powergremlin/

[2] *Instruments* is a performance tool developed by *Apple* integrated tightly with *Xcode*.

Program 3.1: The code used to load the private API into the project. The complete file can be found in Appendix A.

```
1 void PG_loadOSDBattery (void) {
2     NSBundle *b = [NSBundle bundleWithPath:@"/System/Library/
      PrivateFrameworks/GAIA.framework\"];
3     if (![b load]) {
4         @throw [NSError errorWithDomain:@"com.palominolabs.
      UnableToLoadGAIA" code:0 userInfo:nil];
5     }
6 }
```

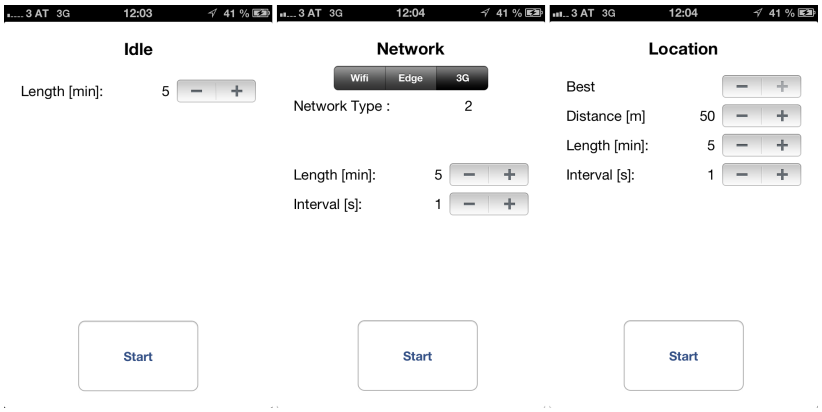


Figure 3.1: The three views created for each of the preliminary tests.

Due to the older model of the iPhone, not only the battery design capacity is lower, but as the phone was used, the battery capacity decreased over time. Just as any battery using anodes and cathodes, the maximum capacity slowly degrades with each charge cycle. Therefore the previously mentioned tool *Powergremlin* can read design capacity as well as the current maximum capacity and the current capacity—each measured in milliampere-hours (mAh)—of which the last is nothing else than the current battery level which is displayed for the user in percent on the right side in the status bar of her phone. Before the tests, the values of the test phone right after charging and unplugging it are as follows:

- design capacity [mAh] 1024,
- maximum capacity [mAh] 945,
- current capacity [mAh] 925.

3.2 Idle State

In order to have a basis for comparison, the idle state energy draw is measured first. Also, some steps need to be taken for the results to be authentic and unbiased:

1. turn off "WiFi",
2. turn off "Bluetooth",
3. turn off "Personal Hotspot",
4. turn off "Location Services", including "System Services",
5. restart the phone.

The first four steps are necessary, as the phone shouldn't use energy for any communication while testing its idle state. The iPhone's "Personal Hotspot" is nothing other than a WiFi hotspot on the phone, tethering its Internet connection. The last step needs to be taken in order for all background processes to be terminated. One could quit one background app at a time, however it is not certain the operation systems isn't running background processes of its own. Restarting the phone is definitely more thorough.

The duration of each test run—idle state as well as the following—is fifteen minutes. The length of the test is necessary to be able to clearly distinguish between a current energy fluctuation and an overall trend. Also, the screen is locked (i.e. turned off) right after starting the test so that it does not influence the results.

The results of the measurements when using *Powergremlin* have one drawback. The value for capacity which is retrieved through the private framework is an integer, influencing all further floating point calculations. However, by increasing the length of the tests this flaw gets more and more insignificant.

Figure 3.2 shows that the iPhone 4 running in idle mode uses about 8 mAh every 15 minutes. Considering the maximum capacity of the phone stated in Section 3.1, that would mean the phone would run out of battery in close to thirty hours. These measurements have been taken with cellular connection 3G enabled and WiFi disabled. The second test run shown in Figure 3.2 has been taken with the phone in plane mode. This has a clear effect on the energy consumption. Because of the discrete value of the capacity however, it would take a longer test run to figure out how much energy this mode exactly requires, but that lies outside the scope of this thesis.

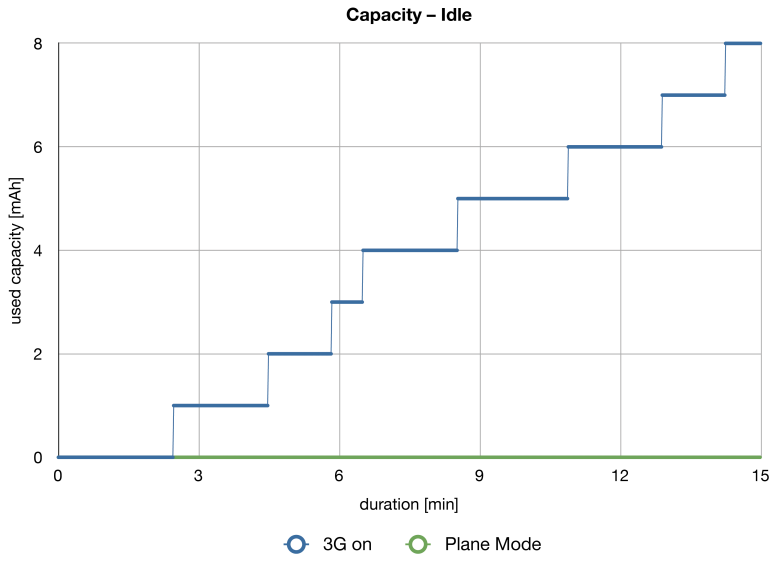


Figure 3.2: Accumulative results of the test when the phone is running in idle mode. Used capacity (mAh) over the duration of the test in minutes.

3.3 Server Communication

The solution proposed in Section 1.4 relies on user data which is fetched from a server. Therefore the energy consumption of server communication is highly relevant. If the cost is too high it renders the proposed solution useless, if the aims of energy-efficiency are not met. Also, the connection type is significant, as it has an effect on the energy requirements as well. The tests for this are prepared the same way as for the idle state, except turning the necessary technologies on and off, respectively.

Figure 3.3 shows the results of the network test runs, continuously sending HTTP-requests for the duration of 15 minutes. A WiFi connection is significantly more energy-efficient, consuming not much more capacity than the idle state during the 15 minutes of the test. The cellular connections draw considerably more power. With this rate of 50 mAh every 15 minutes, the phone would run out of battery in less than 5 hours. The Edge connection is consuming a little less power, but still weighs heavily on the battery.

However, these tests have been designed to place a request every second, which is highly uncommon in normal applications. Therefore, a second round of tests show how also request frequency influences energy usage considerably (Figure 3.4). There is no significant difference between sending a request every second or every minute, as the connection has almost no pause between them. When only requesting every 5 minutes however, the energy cost drops rapidly.

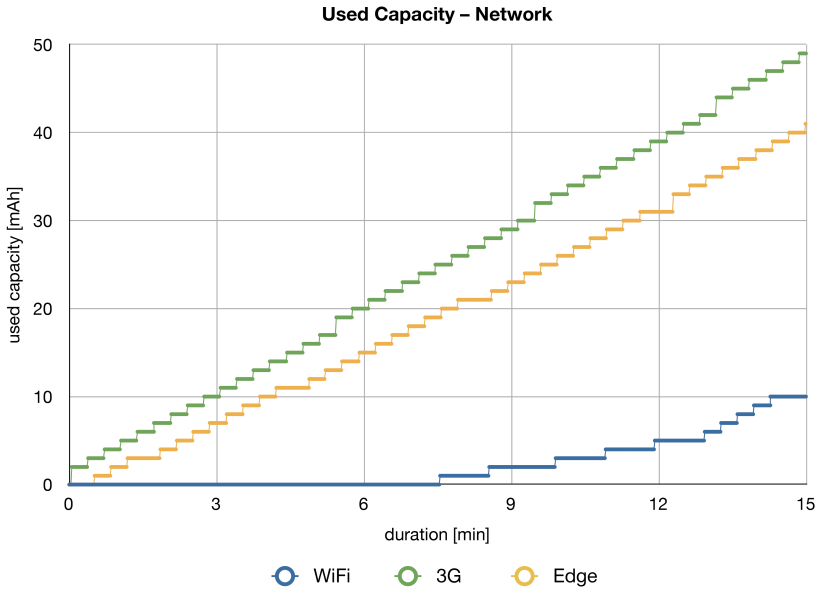


Figure 3.3: Used capacity during network tests with WiFi, 3G and Edge.

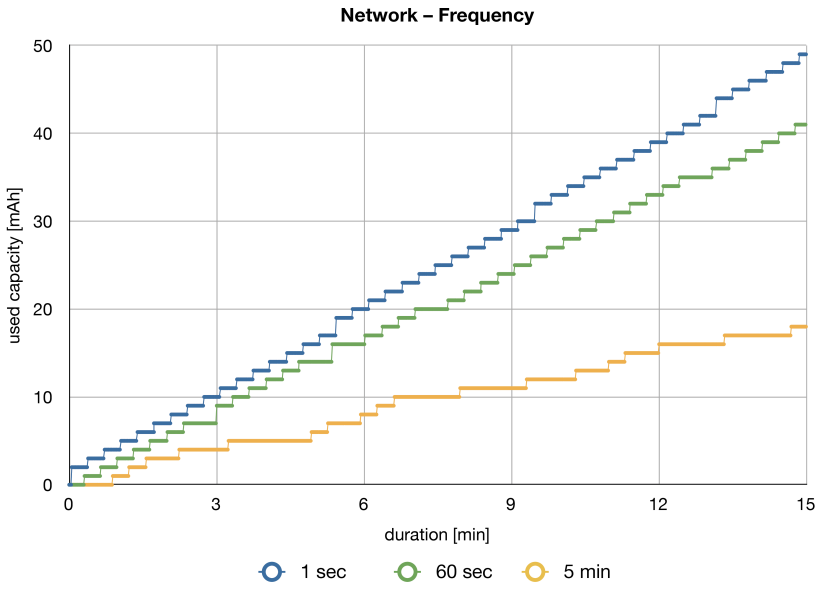


Figure 3.4: Used capacity of 3G network tests at different request intervals.

3.4 Location Sensors

The most important tests in this chapter are for the location sensors. The *iOS SDK* offers location services in the framework *Core Location*. The inner workings of *Apple*'s frameworks are not documented publicly, which makes it even more intriguing to gather the results of these tests. Using the framework *Core Location* it cannot be specified which technology should be used to get the location. Only the desired accuracy can be set. The six different possible accuracies are as follows:

```
1 extern const CLLocationAccuracy kCLLocationAccuracyBestForNavigation;
2 extern const CLLocationAccuracy kCLLocationAccuracyBest;
3 extern const CLLocationAccuracy kCLLocationAccuracyNearestTenMeters;
4 extern const CLLocationAccuracy kCLLocationAccuracyHundredMeters;
5 extern const CLLocationAccuracy kCLLocationAccuracyKilometer;
6 extern const CLLocationAccuracy kCLLocationAccuracyThreeKilometers;
```

Unfortunately, there are no official documents describing what technologies lie behind these steps, but in best case scenarios cellular localisation is the least accurate, followed by WiFi and then GPS as the most accurate. However, it is more prudent to refrain from theorising about the sensors at this point and wait for the outcome of the tests.

One major difference when conducting the tests is that in order for it to work, the phone must be on the move throughout the test run to trigger location updates and keep the sensors active. Also the surroundings should vary, because WiFi localisation accuracy, for example, varies greatly when switching from urban to rural areas. Therefore all tests are conducted at about 15–20 kilometres per hour in rural as well as urban areas. Also—unless stated otherwise—WiFi is turned on in order for it to work as a location sensor as well.

3.4.1 Urban Areas

The results of the first location sensor tests, which are presented in Figure 3.5, are unexpectedly similar to one another. Furthermore, the measurements for the least accurate setting of three kilometres is shown to be the most energy consuming, even if it is not profoundly higher. These results contradict not only the assumptions for this thesis but also the arguments and results of other work like [7] and [4]. To be able to understand the reasons behind these results, more tests need to be made.

Due the fact that in urban areas the amount of WiFi networks is substantially larger, the following test is taken with WiFi turned off, disabling it as a location system. The results, presented in Figure 3.6, show that it still does not make much difference either way. In fact, when comparing the different accuracies with WiFi disabled, as shown in Figure 3.7, the lowest accuracy setting again has the highest energy cost.

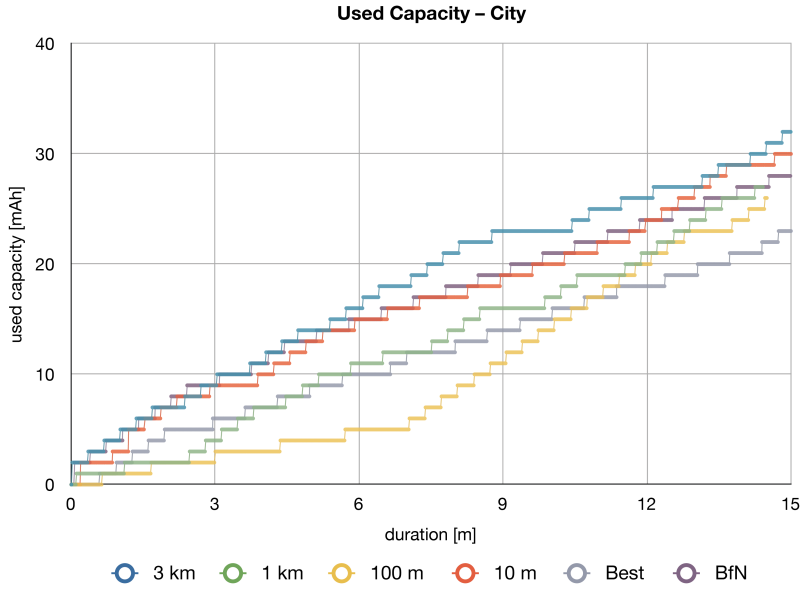


Figure 3.5: First results of the capacity tests for localisation in the city of Linz, Austria. Six different tests—one for each accuracy available in the *Core Location* framework.

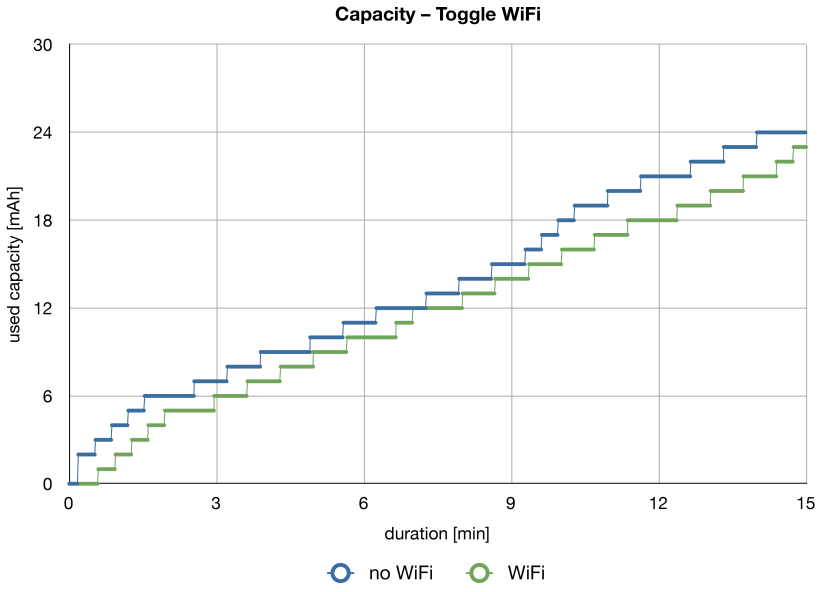


Figure 3.6: Comparative graph showing the used capacity of the test using accuracy `kCLLocationAccuracyBest` once with WiFi enabled, once disabled.

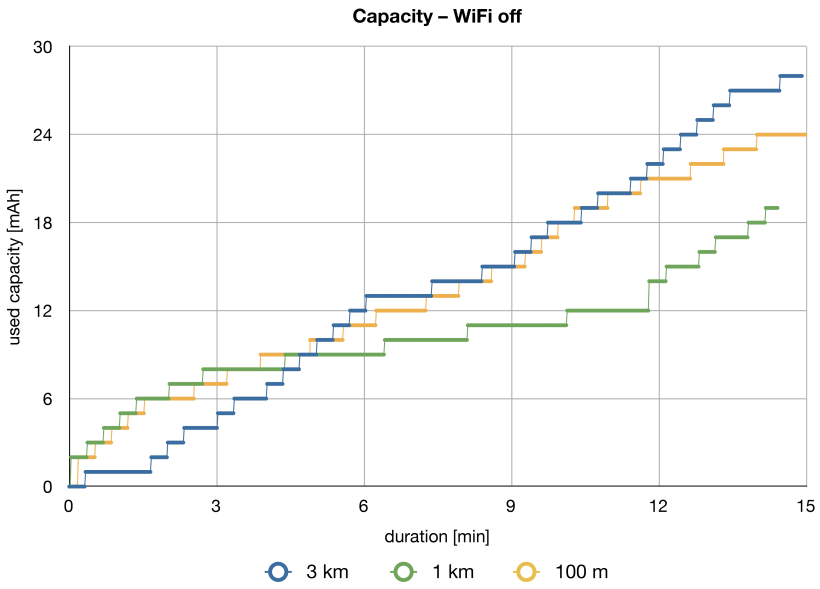


Figure 3.7: Used capacity when disabling WiFi at 3 different accuracies.

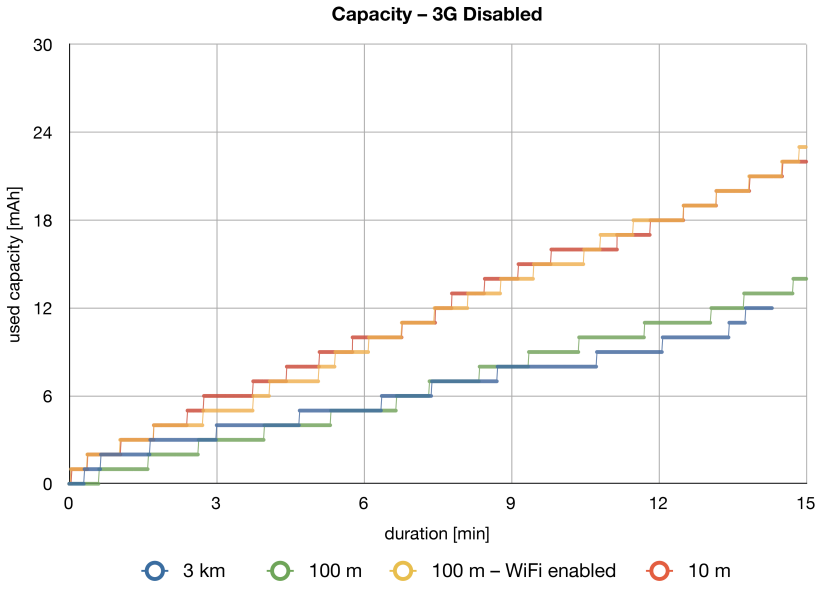


Figure 3.8: Used capacity tests with 3G disabled in the cellular settings of the phone. WiFi was disabled for the accuracies of 3 kilometres and one of the 100 metres.

The lowest accuracy setting is considered to make use of cellular localisation. If this has such a high energy consumption, the reason for these unexpected results must lie with the cellular connection. Therefore another series of tests is made, this time disabling 3G in the cellular settings. As the results show in Figure 3.8, the difference in energy consumption is now visible. With WiFi disabled, the desired accuracies of "3 km" and "100 m" have similar energy cost as well as "100 m" and "Best" have very similar consumption readings. One might come to the conclusion that the technologies and sensors used are the same as well, however when comparing their actual accuracy during the tests, a more informed statement can be made. The framework *Core Location* provides not only the locations, but also the estimated actual accuracy of these. The comparison is shown in Figure 3.9. In this graph one can see that for the desired accuracies of "3 km" and "100 m" with WiFi disabled, the uncertainty is very much the same at around 1400 metres. The desired accuracies of "100 m—WiFi enabled" and "Best", however, are different from each other, supporting the assumption that GPS is used in the latter.

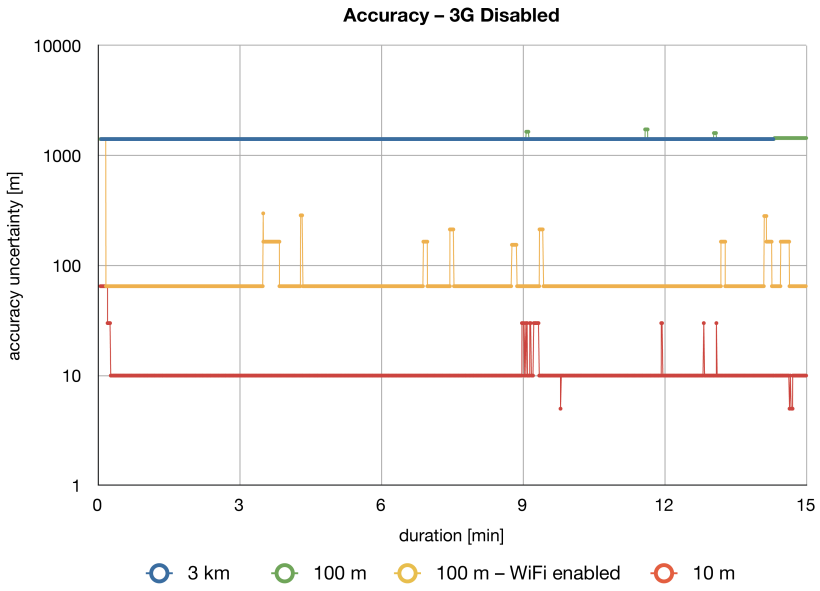


Figure 3.9: Accuracy uncertainty of the tests with 3G disabled on a logarithmic scale. The measurements "3 km" and "100 m" overlap, therefore only one is visible for the most part.

In comparison to these results of the iPhone 4, 3G does not have as much influence on the energy measurements on the iPhone 5, where the different desired accuracy settings are distinguishable through their energy cost even when 3G is enabled, as shown in Figure 3.10.

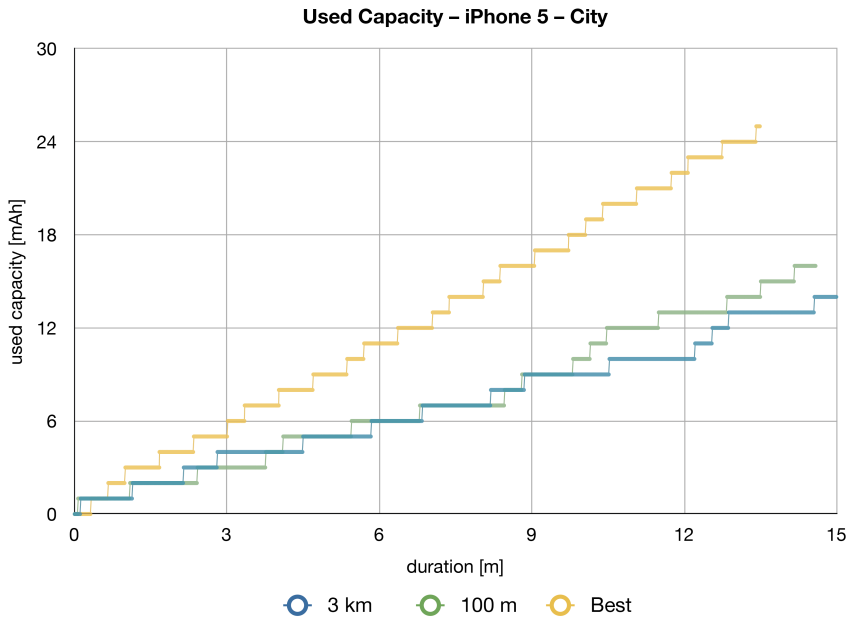


Figure 3.10: Results of the capacity tests on the iPhone 5 in the city with 3G enabled.

3.4.2 Rural Areas

Due to the almost complete absence of WiFi networks and less availability of 3G connection, the tests in rural areas show a much more intelligible result. Figure 3.11 shows that ultimately, every increased step of desired accuracy increases the energy cost. The only exception here being the "10 m" which in this case uses more energy than "Best" and "BfN".

The actual accuracy of the different settings in rural areas is presented in Figure 3.12. Similar to the accuracies in the city, there are 3 main regions which the plotted lines surround. Each of the traces remain at about 1500, 200 or 10 metres for the most part of the test. The lines "10 m", "Best" and "BfN" (best for navigation) all fluctuate around 10–5 metres uncertainty, supporting the assumption that with these settings GPS is involved. The other traces are quite uncertain, only getting more accurate starting minute 6—which was the time when the test route went past a few houses for the first time, supporting the assumption that WiFi localisation is taken into account, if available.

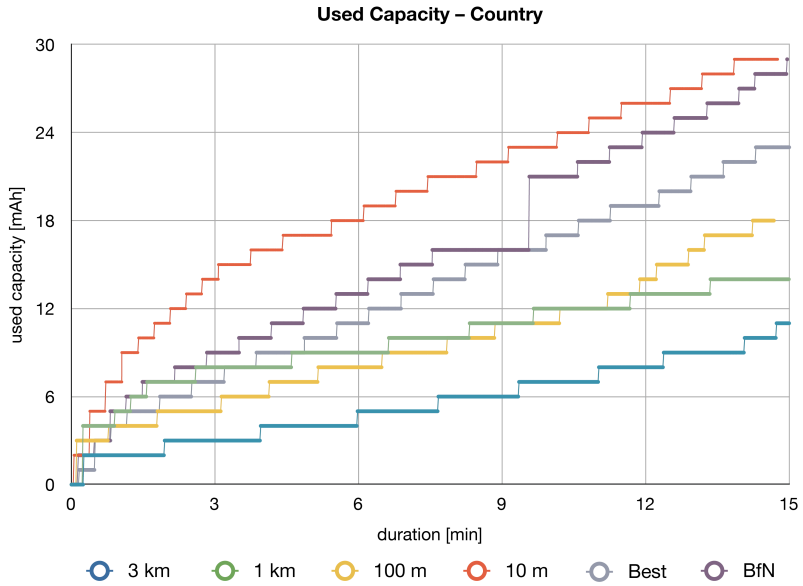


Figure 3.11: Accumulative results of the capacity tests for localisation in the country, near Hagenberg, Austria. WiFi and 3G are enabled. Six different tests—one for each accuracy available in the *Core Location* framework.

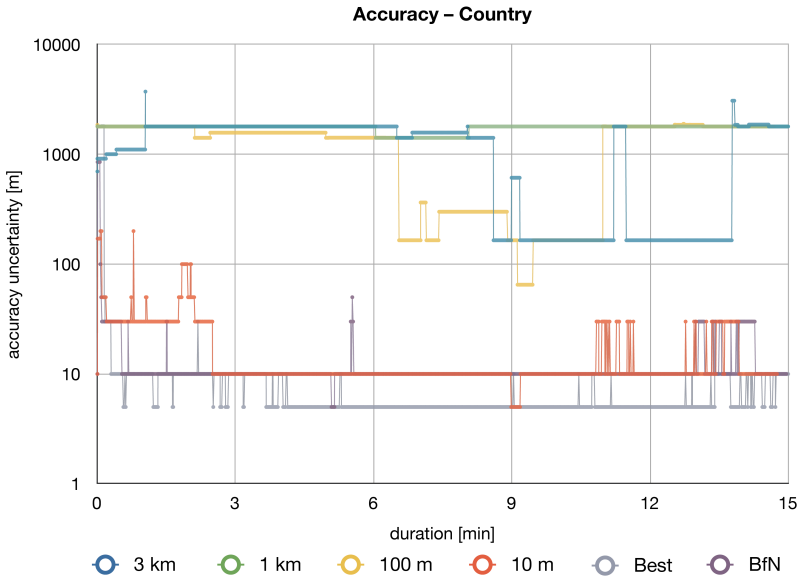


Figure 3.12: Actual accuracy uncertainty, as provided with each location by the *Core Location* framework, on a logarithmic scale.

3.5 Evaluation

It has proven to be challenging to measure energy consumption on the iPhone. This is mainly due to the fact, that all readings are taken from the battery in general, making it difficult to control what contributes to the energy cost. The tool *Powergremlin* is quite powerful due to its use of private APIs, however it has the disadvantage of discrete measurements.

Nevertheless, some relevant findings could be gathered. Firstly, when the phone is running in idle mode, it uses about 8 mAh every 15 minutes. This provides a reference for all other measurements to compare to in order to evaluate energy-efficiency.

Concerning network connection, there is a clear gap between WiFi connections and cellular connections. WiFi uses only a fraction of the energy of 3G or Edge. However, when comparing the energy usage of HTTP-requests at different request intervals, one can see that server communication is not disconcertingly influential to the overall cost, if it is not used continuously.

For location, the biggest differences of energy cost are between urban and rural areas. Using the same settings—WiFi and 3G enabled—the used capacity is mostly proportional to the desired accuracy, except for the iPhone 4 in the city, where it is always high. As later tests and Figure 3.8 reveal, this is due to the 3G network. The 3G network has a high energy toll in urban areas. Also significant for the further course of the thesis is, that up to the desired accuracy of 100 metres, the used capacity is proportional to it. At increased desired accuracies the order is not so clear any more, however the actual accuracy increases greatly, as shown in Figure 3.12.

Chapter 4

Letterbox

4.1 Idea

The project *Letterbox* emerged during the development of a location-based mobile game, when during implementation the thought of being able to leave messages for other players within the game was toyed with. That simple idea was later picked up and led to the initiation of *Letterbox* as a separate application.

The thought was to create a communicative mobile application based on the location of the users. The "Letterbox" should just be a digital version of a letterbox, tied to a specific position by coordinates. Every user should be able to create her own and be able to leave messages—"letters"—in others.

The idea quickly evolved and a second way of communication was added. Not only should letters be placed in boxes, but work as personal messages as well, placing them at specific locations for other users to find. This turned out to become the main part of the concept to make letterbox a "location-based messenger".

4.2 Concept

The service of *Letterbox* needs two main components—a mobile client application and a server which handles communication. They should provide the following tasks:

- write digital letters and leave them at specific places for specific people,
- create digital letterboxes at places where letters can be submitted,
- set privacy—define who is allowed to read what,
- notifications—prompt the user when a message for her is nearby,
- user management and interaction is handled by the server.

One key aspect of *Letterbox* is its focus on the content. The app itself should be rather subtle, fading in the background while the content is presented. This should not only be part of the design choices, but influence the app's behaviour at the same time. As long as no new information is relevant for the user, it fades into the background. When a letter for her is nearby, only then is she made aware of it by a notification.

The central element of the concept from the start was energy-efficiency. The fact that the app should prompt the user to collect nearby letters means that location services would need to run in the background continuously. As previously mentioned, this can have a drastic effect on the battery life of the phone.

4.3 Relevance

The relevance of *Letterbox* for this thesis lies in its need for an energy-efficient location-based system architecture in order to not substantially reduce the phones battery life. Furthermore, the use of location-dependent content—the letters and letterboxes—is a good example for the kind of applications introduced in Chapter 1. By focusing on the content relevant to the user, more ways of saving energy are possible.

As many location-based applications, *Letterbox* manages the users online on a server, opening up communication between them. Instead of the user-relevant content of favourite locations and restaurants, as with *Yelp* and *Foursquare*, it is user-relevant messages which are sent to the phones for presentation. The necessary architecture for letterbox could, however, be relevant for all of these applications, should they wish to have continuous location tracking in the background.

4.4 Implementation

Both server and client have been implemented simultaneously, adding features on both ends and keeping the need for energy-efficient design in mind. Where possible, calculations and heavy operations were moved to the server, network communication was kept at a minimum and for energy-efficient location updates, an architecture was created to handle it.

4.4.1 Server

The server in *Letterbox* on the one hand serves as the user management system and on the other is responsible for the administration of the content. Also the authentication process is handled by the server. It is a PHP-application, created with the framework *Symfony*[1].

[1] http://symfony.com/

User Management

To be able to verify the identity of the user, client and server use an authentication process. The app has two codes—an app ID and an app secret, which are sent alongside the registration of a user to verify the identification of the app. The server then generates an access-token, which is stored with the newly registered user and sent back to the client. The token is stored there as well and will from now on be sent alongside every request to the server in combination with the user ID, authenticating it (see Figure 4.1 for a sequence diagram of the process). This has the advantage, that the data can only be accessed by registered users and only via a client. Furthermore, this works as a basis for an API. In the future, more applications could gain rights and access to the server to dynamically expand the *Letterbox* service. Also, users will be able to restore their messages and other content when losing or switching their device. Once registered, the users are managed by the server as well. All communication between the users as well as contact information of each user are managed and exchanged by it.

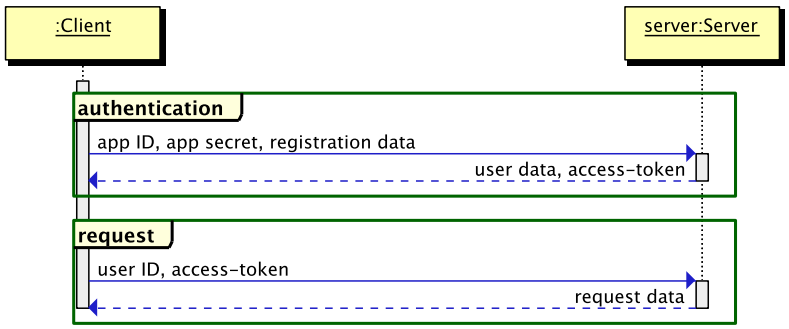


Figure 4.1: Authentication process—app ID and app secret are needed to gain access-token for requests.

Content Administration

Also stored by the server are all messages, letterboxes, privacy settings and the different states of messages (e.g. read or unread). As mentioned before, this has the advantage that the user has access to all her data after the first login on a new device. Also contacts are stored with each user, which are added via contact requests from the client. To find new contacts, a search for names of email addresses is available.

Symfony

The choice of *Symfony* as the base framework has been made, because the data model can be set up quickly and the object-oriented access to it is quite

lucid. In addition, the routing system enables to easily create an expressive API. The predefined JSON conversion methods of the data model objects make it even more suitable to put them into a format which is convenient for the client to interpret.

Client-Server Communication

The client sends its requests according to the API to the server and places the relevant data in POST parameters to make it easier for it to extract them. As a response, the server converts the requested data into JSON format and returns it to the client.

4.4.2 Client

As mentioned in Section 4.2, the client's intent is more about the content than its own design. This led to the client using mainly standard elements of the *UIKit* of the *iOS SDK* for the user interface which are familiar to every iPhone user and do not distract her from the content. The user interface of the app is split into 3 screens (see Figure 4.2):

- Contacts: a list of all stored users,
- Map: all important messages are presented at their location,
- Inbox: the collection of all sent and received messages.

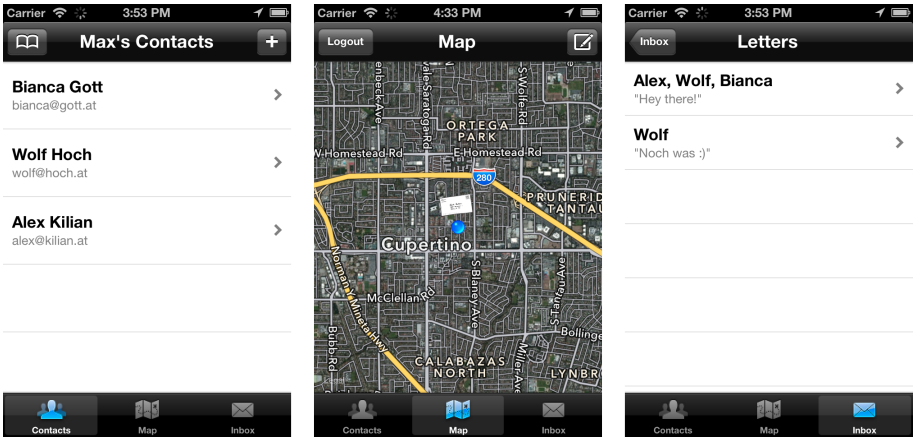


Figure 4.2: The 3 main views of *Letterbox*. Contacts, Map and Inbox.

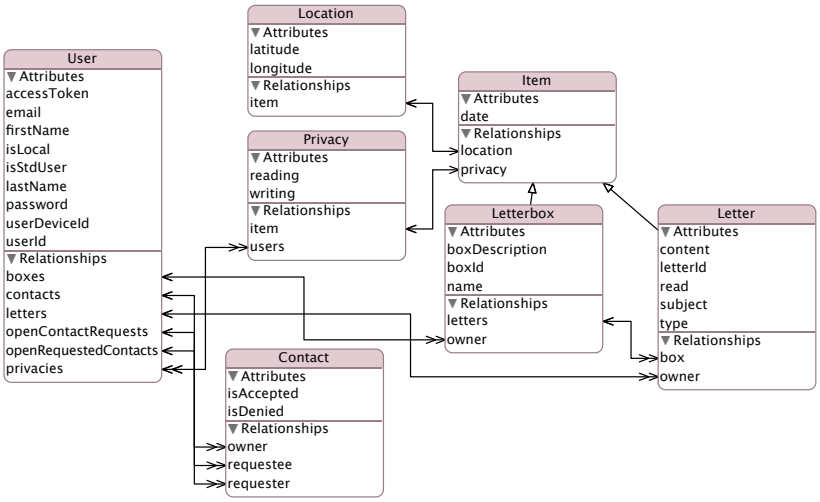


Figure 4.3: The structure of the client's database.

Contacts

In this view, all contacts are listed. To make it easier for the user to search for friends using *Letterbox*, the button at the top left will compare all email addresses saved in the contacts of the iPhone to *Letterbox* users and offers the user to add the matches. A simple textual search is available as well (see left part of Figure 4.2).

Map

The map as an interactive element was implemented with the help of *Map Kit*, an *iOS* framework. It offers classes and protocols to integrate map data provided by *Apple*. As shown in the centre of Figure 4.2, it is a hybrid map of satellite and street views, on top of which new messages are visualised at their specific location. Also the current position of the user is shown. She can interact with the content directly over this view. A new message or letterbox is created by tapping the button to the top right.

Core Data

To implement the data model in the client, the database system *Core Data* was used. Its is fully integrated with *Apple*'s development environment *Xcode* and therefore easily created and used. Furthermore it has the advantage, like the server has with *Symfony*, that the entities are handled as objects in the code, which simplifies the reading and writing of data. The structure of the database is presented in Figure 4.3.

Program 4.1: Server request methods of the `ServerModule`. The one used by the architecture is on line 7.

```
1 /* Helper Methods */
2 - (void)getServerDataAfterUserLogin:(User *)user;
3
4 /* Request Methods */
5 - (void)registerUserWithFirstName:(...);
6 - (void)loginUserWithEmail:(...);
7 - (void)getSurroundingForLocation:(...);
8 - (void)getAllBoxesForUser:(...);
9 - (void)getAllLettersForUser:(...);
10 - (void)getReadLettersForUser:(...);
11 - (void)markReadLetterForUser:(...);
12 - (void)createBox:(...);
13 - (void)createLetter:(...);
14
15 /* Contacts */
16 - (void)searchUsersForUser:(...);
17 - (void)matchUsersWithEmailAddresses:(...);
```

ServerModule: Operation-Queue

The server module, which handles all communication with the server, is essentially an operation queue and consists of multiple interacting classes. Figure 4.4 gives an overview of the structure of the classes involved.

The `ServerModule` class is a singleton, which makes it accessible from everywhere within the application. Also it is safer to only have one instance of the class so that the server responses are collectively handled and forwarded by one object to avoid inconsistency. The `ServerModule` has a set of methods (see Program 4.1) to initiate the server requests. Also, it stores the `NSOperationQueue`, where each request is enqueued in form of an operation.

`ServerOperation` is a subclass of `NSOperation`, which is part of the *iOS SDK*. It again has subclasses which each represent a server request (see Figure 4.4). On instantiation, the URL and POST-parameters are set, which vary with each request. After setting the user ID and access token, the object is added to the queue and awaits its turn to execute its `main`-method, implemented in the superclass `ServerOperation`.

4.5 Architecture

The most important goal trying to be achieved with *Letterbox* is energy-efficiency. The application constantly needs information of its position to be able to inform the user when she passes a location where a message was left for her. The idea for the solution of this problem, as introduced in Section 1.4,

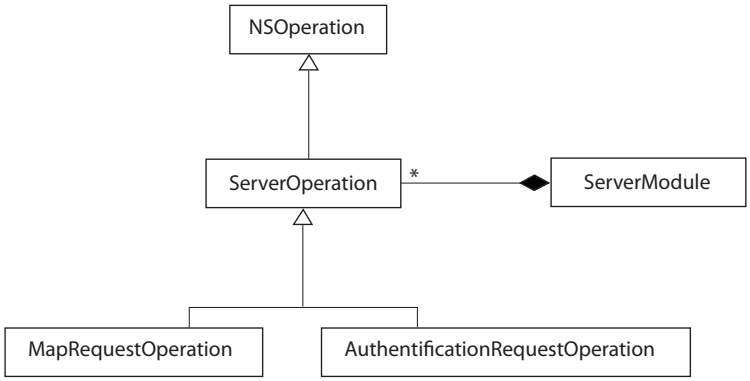


Figure 4.4: The classes of the `ServerModule`.

is to continuously adapt the location accuracy to user-relevant data. With *Letterbox*, this data is the messages and letterboxes users place in its virtual world and send to each other.

The reason for it to be called an architecture is that this solution requires a software system over multiple platforms, involving multiple classes each having specific properties and relations to one another in order for it to achieve the goal of energy-efficiency. The details of this architecture will be presented during the course of this chapter.

4.5.1 Assumptions

The architecture is built on three fundamental assumptions, of which some have been supported by the preliminary tests in Chapter 3 as well as by related work described in Section 2.4. These three assumptions are that energy-consumption on the iPhone is proportional to the desired accuracy steps of the *Core Location* framework, that the energy cost of the necessary network connection is smaller than the energy gain through the use of the architecture and that by focusing on user-relevant location-based data the energy-efficiency can be increased considerably.

Proportional Energy

The main assumption is that by dynamically changing the desired accuracy of the location services, the energy consumption changes with it proportionally. Similar to this, the authors of [7] have argued that their approach is able "to determine the most energy efficient sensor to be used, such that the required location accuracy can be achieved." The difference to this architecture is that it determines the desired accuracy best suited for localisation, such that

energy cost is low and accuracy at the relevant positions remains high. This is not possible on the iPhone with the *Core Location* framework, therefore it is assumed that the energy cost is also proportional to the desired accuracy steps. Chapter 3 shows that this assumption is correct, but has limitations. In rural areas the desired accuracy steps all correspond to the energy cost, except for the “10 metres" setting. In the city, the energy consumption is highly influenced by the 3G network connection on the iPhone 4. Only when disabling it, some differences in energy cost can be found as well. Most clear are the steps “3 kilometres”, “100 metres” and “best” which have thus been selected to be used in the architecture.

Network Energy Cost

As the architecture relies on user-relevant data from the server, it is essential that network requests do not use too much energy. If the energy required to get the information needed from the server is equal or higher than the energy saved by the use of the architecture, it renders the whole concept ineffective.

The preliminary tests show, that a 3G connection needs more energy than WiFi. Unfortunately, it cannot be influenced by the architecture what connection type is used. More relevant for the design is the length of the intervals between the requests. When building the architecture, it is crucial to make it have as little network activity as possible.

Location-Based Data

One of the key points of the architecture is the focus on location-based content to reduce energy consumption. The idea is that accurate localisation is only necessary when being near such content. The density of the content is therefore the determining factor. If it is constantly too high, the architecture is not effective. Hence the architecture is more useful, if the number of location-based content which requires accurate positioning around it is kept within reasonable limits.

4.5.2 Key Features

Based on these assumptions, the following key features of the architecture have been designed. An experiment should later determine the actual values of the parameters involved in:

- server-client communication,
- dynamic adjustment to desired accuracy,
- consideration of current actual accuracy,
- focus on location-dependent content.

Server-Client

For the server-client communication, the goal is to have as little activity as possible. Thus when pulling the new messages from the server, the scope is set to 20 kilometres. This means that a rectangle defined by latitude and longitude coordinates is set with a distance of 10 kilometres in every direction and sent to the server when requesting new messages, if available. This ensures that the need of another update only exists when moving these 10 kilometres, assuming no new messages arrive for the user in that time. The value of 10 kilometres is chosen to be big enough for the user to not miss any content—she would need to go at 60 kilometres per hour to do so—and small enough to limit the size of the content returned in the response.

By considering Figure 3.4 and after a process of trial and error, the interval in between updates is set to 10 minutes, ensuring a low energy consumption level. The disadvantage at this point is that new messages may have a delay of up to 10 minutes when being put in the vicinity of the receiving user. A possible way to counter this problem would be to implement push notifications for new messages. This might bring about more performance in *Letterbox* specifically, but is not part of the architecture and therefore not yet implemented.

Location Dependent Accuracy

Due to the findings of the preliminary tests, three steps of the desired accuracy setting have been selected to dynamically switch in between:

- "3 km"—for tracking the location if the distance to the nearest relevant content is at least 3 kilometres,
- "100 m"—if the distance to the nearest relevant content is in between 3 kilometres and 100 metres,
- "Best"—when the distance to the nearest relevant content is less than 100 metres.

These 3 desired accuracies have the most noticeable difference in energy cost throughout most preliminary tests from Chapter 3. In the city—as long as 3G is disabled on the iPhone 4—as well as rural areas the difference in energy consumption is visible. The distances to the nearest relevant content have been chosen as a first naive approach in order for the desired accuracies being enough to assure that the user has not yet arrived at the content (outside of 3 kilometres, "3 km" accuracy would be sufficient, outside 100 metres, "100 m" accuracy would be enough, etc.).

There is however one flaw in this system. The accuracy that can be set in the *Core Location* framework in the *iOS SDK* is the `desiredAccuracy`-property. It is not the minimal required accuracy, but only the desired

accuracy. If it is not available, greater inaccuracies are possible. This can be seen when looking at the results of the preliminary tests in Figure 3.12. The yellow "100 m" line is, for the most part, over 100 metres—even over 1000 metres—as mostly only cellular localisation is available and GPS is not yet activated at this setting. This means, that the actual accuracy needs to be taken into account when deciding on which settings to use.

To consider the actual accuracy can have a positive effect on energy-efficiency as well. If, for instance, the user passes the 3 kilometre boundary to the relevant content, usually the next desired accuracy step is selected. If, however, the actual accuracy is already higher than the desired "3 km", this action can be postponed until the user is closer to the content than the current location's accuracy uncertainty. This could bring about a lower energy cost, as it might reduce the time using expensive location sensors.

4.5.3 Implementation

The implementation of the architecture on the client was capsuled in the singleton class `LocationModule`. Just as the `ServerModule`, it is a singleton to make sure that only one instance is responsible for location tracking and deciding which level of accuracy is appropriate. The localisation procedure is visualised as a flow diagram in Figure 4.5.

LocationModule

The `LocationModule` owns an instance of the `CLLocationManager` and implements the necessary methods of the `CLLocationManagerDelegate`-protocol. It is responsible for all things concerning location tracking. It defines three macros to distinguish between the desired accuracy steps:

```
1 #define LOCATION_ACCURACY_FAR  kCLLocationAccuracyThreeKilometers
2 #define LOCATION_ACCURACY_MEDIUM  kCLLocationAccuracyHundredMeters
3 #define LOCATION_ACCURACY_CLOSE  kCLLocationAccuracyBest
```

In the `init`-method, after initialising the properties, the desired accuracy is set to `LOCATION_ACCURACY_FAR` and the `CLLocationManager` is started by calling its `startUpdatingLocation`-method. At this point the delegate method `locationManager:didUpdateLocations` is called whenever a new location is available, giving the architecture the chance to respond to it. In the `LocationModule`, the method `checkDistancesToLocation` is called to further handle the event. The code is shown in Program 4.2. In this method, the smallest distance to a user-relevant location is determined. Then, starting at line 28 of the code excerpt, the new accuracy level is determined based on this distance. At last—at line 36 and 42—it is determined whether the accuracy level should be increased or can be decreased when further taking the actual accuracy of the current location into account.

Program 4.2: The `LocationModule`'s `checkDistancesToLocation`-method handls each new location.

```
1 - (void)checkDistancesToLocation:(CLLocation *)location {
2
3    self.smallestDistance = 3000.0;
4
5    for (Letter *l in self.unreadLetters) {
6       CGFloat d = [l.location distanceToCoordinate:location.coordinate];
7       self.smallestDistance = MIN(self.smallestDistance, d);
8       if (d<20. && ![self.nearItems containsObject:l]) {
9          // [...] notify user about new letter nearby
10         [self.nearItems addObject:l];
11      }else if (d>=20. && [self.nearItems containsObject:l]) {
12         [self.nearItems removeObject:l];
13      }
14   }
15
16   for (Letterbox *b in self.unreadBoxes) {
17      CGFloat d = [b.location distanceToCoordinate:location.coordinate];
18      self.smallestDistance = MIN(self.smallestDistance, d);
19      if (d<20. && ![self.nearItems containsObject:b]) {
20         // [...] notify user about new letterbox nearby
21         [self.nearItems addObject:b];
22      }else if (d>=20. && [self.nearItems containsObject:b]) {
23         [self.nearItems removeObject:b];
24      }
25   }
26
27   // initial setting of desired accuracy level based on smallest distance
28   int newAccuracyLevel = 0;
29   if (self.smallestDistance < 100.) {
30      newAccuracyLevel = 2;
31   }else if (self.smallestDistance < 3000.) {
32      newAccuracyLevel = 1;
33   }
34
35   // check if actual accuracy too big for current smallest distance
36   if (self.smallestDistance < location.horizontalAccuracy + 10) {
37      newAccuracyLevel = MIN(newAccuracyLevel+1, 2);
38   }
39
40   // check if actual accuracy was good enough for
41   // current smallest distance and change level back
42   if (newAccuracyLevel > self.currentAccuracyLevel
43       && self.smallestDistance > location.horizontalAccuracy + 10) {
44      newAccuracyLevel = MAX(newAccuracyLevel-1, 0);
45   }
46
47   [self setAccuracyLevel:newAccuracyLevel];
48 }
```

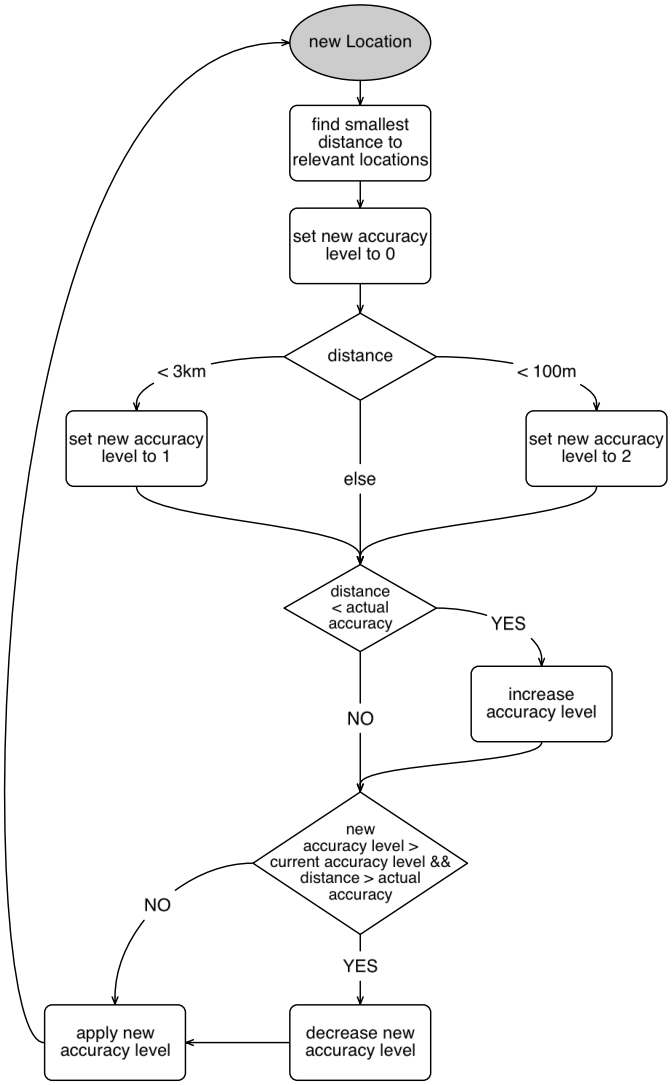


Figure 4.5: The process of handling a new location update in the architecture. The accuracy levels are 0 for "3 km", 1 for "100 m" and 2 for "Best" desired accuracy.

In the last line of the `checkDistancesToLocation`-method, the call to set the new accuracy level is made. In this method, the desired accuracy is determined based on the accuracy level and then the following lines of code are executed:

```
1   self.locationManager.desiredAccuracy = newAccuracy;
2   self.locationManager.distanceFilter = self.smallestDistance * 0.33;
```

The desired accuracy is being handed to the location manager, and the `distanceFilter`-property is set. The distance filter is the minimum distance in meter the location has to change in order for a location update to be triggered. It is set to be a third of the way to the nearest relevant content. This means that the closer the user gets, the more often the position will be checked and updated. It ensures more accurate tracking at relevant positions and more energy being saved everywhere else.

DBModule

The database module on the client is responsible for all the content of *Letterbox*. For the architecture, this means being responsible for the content updates. It contains the timer which initialises the requests to the server every 10 minutes. And more importantly, it manages the response from the server, filtering for new messages and other content which will be regarded as relevant for the `LocationModule`.

ServerRequestSurrounding

Every 10 minutes, the application pulls the new messages in the area for the user within 10 kilometres in every cardinal direction. In the `init`-method of the `ServerRequestSurrounding` class, shown in Program 4.3, the distances for latitude and longitude need to be converted separately into degrees, because the distance for a longitude degree varies according to the current latitude. The distances are calculated with the formulas

$$offset_{latitude} = \frac{111}{kilometres} \tag{4.1}$$

and

$$offset_{longitude} = \frac{kilometres \cdot \cos(position_{latitude}) \cdot 6378}{(2 \cdot \pi/360)}. \tag{4.2}$$

4.5.4 Expected Limitations

The purpose of the architecture is to increase energy-efficiency. However, there are some expected limitations of the process. One of these is that if the number of user-relevant location-based objects and hence the density of them increases, the energy usage in that area will increase as well. If the density surpasses one relevant object per 100 m^2, the energy usage will be the same as when constantly using the desired accuracy "Best".

Another limitation is that in the city, energy-saving is most effective in general if initiated by the user—not purely through the architecture. Due to the high energy consumption of 3G in urban regions, as shown in Chapter 3, the resulting energy saved is considerably less compared to when the user

Program 4.3: The `init`-method of the `ServerRequestSurrounding` class. Latitude and longitude distances need to be calculated separately.

```
1 - (id)initWithCoordinates:(CLLocationCoordinate2D)coordinate
2                   accuracy:(CLLocationAccuracy)accuracy
3                       user:(User *)user
4 {
5     self = [super init];
6     if (self != nil) {
7         float lngDistance = [LocationTools
8                              getLngDistanceForKilometers:10.
9                              fromCoordinate:coordinate];
10        float latDistance = [LocationTools
11                             getLatDistanceForKilometers:10.
12                             fromCoordinate:coordinate];
13
14        int bottomLeftLat = (coordinate.latitude-latDistance)*1000000;
15        int bottomLeftLng = (coordinate.longitude-lngDistance)*1000000;
16
17        int topRightLat = (coordinate.latitude+latDistance)*1000000;
18        int topRightLng = (coordinate.longitude+lngDistance)*1000000;
19
20        NSString *post = [NSString stringWithFormat:@"&user_id=%lld&
      access_token=%@&minimal_latitude=%d&minimal_longitude=%d&
      maximal_latitude=%d&maximal_longitude=%d",
21                          user.userId,
22                          user.accessToken,
23                          bottomLeftLat,
24                          bottomLeftLng,
25                          topRightLat,
26                          topRightLng];
27
28    return [super initWithRequestURL:kRESTServerSurroundingUrl
29                                  method:@"POST"
30                           andPostString:post];
31    }
32    return self;
33 }
```

disables 3G in general. This means that in the city the expected energy saved by the user turning off 3G is bigger than the energy saved by the architecture.

In general, the proposed architecture is therefore only effective for applications with a limited set of significant locations. Also, *Letterbox* is expected to be more energy-efficient with the architecture than without it, however its effect may be limited if the user has 3G enabled, as it is the case on the iPhone 4.

Chapter 5

Experiment

To test the performance of the architecture, an experiment is conducted. The goals are to determine the amount of energy saved by the architecture, the accuracy at the relevant positions and whether through some adjustments to the variables of the architecture, the performance can be increased further. The application *Letterbox* is the basis of the experiment and servers as the container for the architecture.

5.1 Preparations

In order to be able to measure the energy consumption of the architecture, the *Letterbox* app needs to be prepared accordingly. The tool *Powergremlin* is added to the project and implemented into the app. Along with it, the class `TestModule` from the preliminary tests is added to the project and integrated into the code. A new button has been added to one of the view controllers (Figure 5.1) in order to start and stop the measurements in the following method:

```
1 - (IBAction)startTest:(id)sender {
2     if (![TestModule sharedTestModule].currentTestLog.running) {
3         [[TestModule sharedTestModule] startIdleTestWithLength:2400];
4     }else {
5         [[TestModule sharedTestModule] stopTest];
6     }
7 }
```

When the button is tapped, a new “idle” test log will be initiated with the maximum length of 40 minutes. The “idle” test does nothing more than logging the energy readings, as opposed to the “network” and “location” logs, which is why it is used for the experiment.

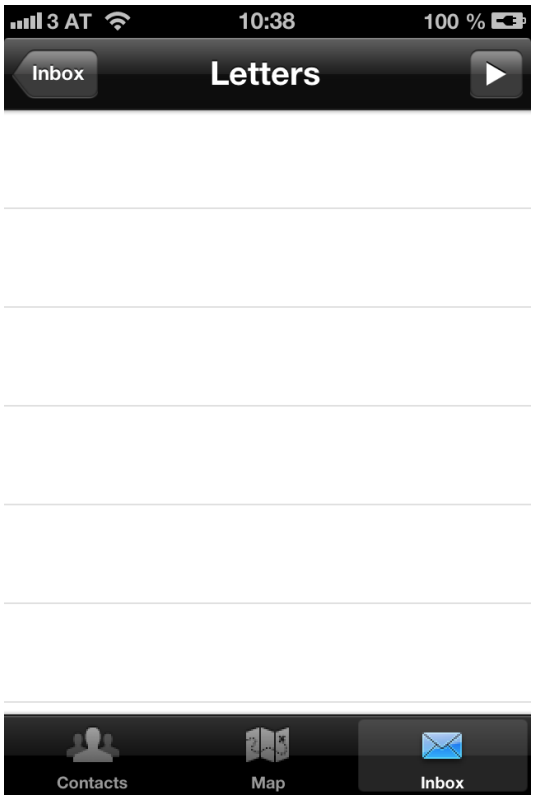


Figure 5.1: The `ConversationsViewController` of the *Letterbox* app including the added test button for energy measurements in the top right corner.

5.1.1 Consistency

To ensure that the *Letterbox* app itself, apart from the architecture, influences the measurements as little as possible, some adjustments to the `LocationModule` have been applied as well. A fixed set of locations is determined to be regarded as relevant:

```
1 self.locations = @[@{@"lat":@48.297472, @"lng":@14.289660},
2                    @{@"lat":@48.298854, @"lng":@14.296153}];
```

The main reason for this is, that the *Letterbox* app uses `NSManagedObjects` for letters and locations, which is the super class for all objects handled by the *Core Data* framework, therefore entailing multiple calculations and database statements in the background which potentially could influence the results. Furthermore, all user notifications are removed to decrease activity in the background on the one hand, but also ensure that the screen does not become active during the test runs.

Program 5.1: The `LocationModule`'s `checkDistancesToLocation`-method after the preparations for the experiment.

```
1 - (void)checkDistancesToLocation:(CLLocation *)location {
2
3     self.smallestDistance = 3000.0;
4
5     for (NSDictionary *d in self.locations) {
6         CLLocationCoordinate2D c;
7         c.longitude = [[d objectForKey:@"lng"] doubleValue];
8         c.latitude = [[d objectForKey:@"lat"] doubleValue];
9         CGFloat distance = [LocationTools distanceFromA:c
10                                                 toB:location.coordinate];
11
12        self.smallestDistance = MIN(self.smallestDistance, distance);
13
14        if (distance < 20 && ![self.nearItems containsObject:d]) {
15            [self.nearItems addObject:d];
16        }else if (distance >= 20 && [self.nearItems containsObject:d]) {
17            [self.nearItems removeObject:d];
18        }
19    }
20
21    int newAccuracyLevel = 0;
22
23    if (self.smallestDistance < 100.) {
24        newAccuracyLevel = 2;
25    }else if (self.smallestDistance < 3000.) {
26        newAccuracyLevel = 1;
27    }
28
29    if (self.smallestDistance < location.horizontalAccuracy + 10) {
30        newAccuracyLevel = MIN(newAccuracyLevel+1, 2);
31    }
32
33    if (newAccuracyLevel > self.currentAccuracyLevel
34        && self.smallestDistance > location.horizontalAccuracy + 10) {
35        newAccuracyLevel = MAX(newAccuracyLevel-1, 0);
36    }
37
38    [self setAccuracyLevel:newAccuracyLevel];
39
40    [[TestModule sharedTestModule]
41          addLogEntryToCurrentTest:location.horizontalAccuracy
42                                  :newAccuracyLevel+1];
43 }
```

The resulting `checkDistances`-method, as first presented in Program 4.2, is shown in Program 5.1. On lines 5–19, the fixed locations are interpreted as relevant positions for the architecture. The rest of the method remains

the same, containing the logic of the selection of the accuracy level. The exception is line 40, where the new log entry of the current measurements is made, including the current actual accuracy and the new desired accuracy level. However, the network updates are still performed every 10 minutes in the background, as it is part of the architecture and needs to add to the overall energy cost.

5.2 Main Setup

The main setup of the experiment is similar to the preliminary tests. The phone is prepared as described in Section 3.2 before each test, which is then conducted as defined in the location tests from Section 3.4. The main differences are the duration of the tests and the length of the test route.

5.2.1 Test Route

The test route and positions of the relevant data for the experiment are selected to meet the following requirements:

- it should at some point have 3 kilometres distance to the nearest relevant location,
- it should be both outside and inside the city,
- it should lead past at least one relevant location at 100 metres distance,
- it should go directly over at least one relevant location.

The resulting route and locations which have been selected are shown in Figure 5.2. The part of the route that forms a closed circuit is the same as the test route of the preliminary tests in Chapter 3.

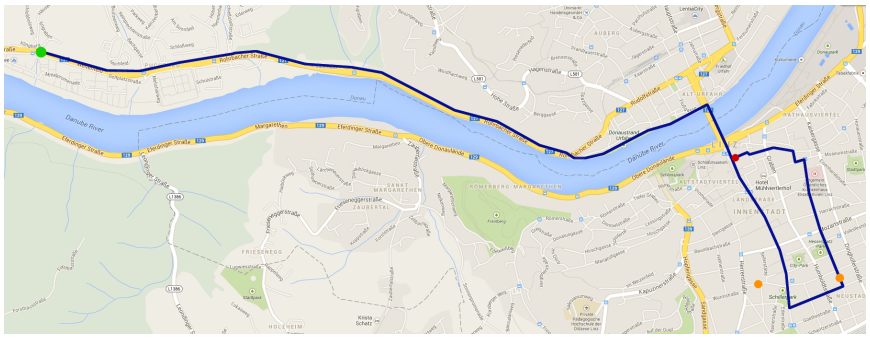


Figure 5.2: The test route for the experiment, starting at the green circle, passing the relevant locations in orange and ending at the red circle. The part of the route that forms a closed circuit corresponds to the test route of the preliminary tests. Map data is taken from *Google* (http://maps.google.com/).

5.2.2 Duration and Speed

Due to the increased length of the test route, the duration of the tests increases as well. This is a necessity, however also has the advantage that overall energy consumption trends are more recognisable. To save time, the speed is about 30–35 kilometres per hour for the first part of the route—up until the crossing of the river—which is approximately double the speed of the preliminary tests. The second part of the route in the inner city is still completed with a speed of about 15 kilometres per hour. The increased speed at the beginning is not expected to cause problems, as it is far from the relevant locations and therefore the accuracy is still low.

5.2.3 Location Accuracy

The test route starts outside the city with the purpose of being more than 3 kilometres from the nearest relevant location. This ensures all desired accuracy steps to be part of the experiment. When getting near the first location, which is about 100 metres next to the test route, it depends on the actual accuracy of the current position if the architecture increases its desired accuracy or not. For the second relevant location, which is directly on the test route, the desired accuracy will have to be at "Best" for the experiment to be successful.

5.3 Variations

With each test run of the experiment, some variations are brought to different aspects of the architecture or the settings of the phone. This provides more knowledge about the effects of the separate parts of the architecture as well as a better understanding of what might increase performance even more or has a negative effect on the results.

5.3.1 Preconditions

As the results of the preliminary test show in Chapter 3, the phone settings have a profound impact on the energy consumption of the phone. Therefore, the tests are conducted once with 3G enabled and once disabled. This will also give indication of how much the user can assist the energy-efficiency of the architecture and if it has any benefit at all with 3G activated. Also, as the energy consumption of the lowest desired accuracy setting surpassed all others with 3G enabled, it can be argued that 3G has an increased effect on energy consumption for location tracking. As the preliminary tests do not indicate that WiFi has an influence on the energy-efficiency, it will remain enabled for all tests.

To gain a basis of comparison, one variation is that the desired accuracy is set to "Best" the whole time. This shows the difference between the energy cost of the architecture as opposed to not using any means of energy saving.

5.3.2 Distance Variables

Another aspect influencing energy-efficiency are the distance values with which the architecture is built. If the user is moving at greater speeds, she might pass through the 100 metre range of the "Best" accuracy setting before the GPS can find an accurate position. However, if as a countermeasure the 100 metre mark—signalling the switch to the highest desired accuracy—is moved out to 500 metres, the GPS would be activated much more often, costing more energy. This would at some point render the architecture useless. To verify these assumptions, test are also made with differences in the distance values.

5.4 Results

Overall, the obtained results are promising with regard to energy-efficiency, however the accuracy readings are disappointing at first sight. Furthermore, the architecture's effects are limited on the iPhone 4 when 3G is active.

5.4.1 Energy Saved

The comparison of the tests on the iPhone 4 is shown in Figure 5.3. The energy consumption when using the architecture is about 58 mAh per hour. This means that the average draw is about 58 milliampere. When using the desired accuracy "Best" constantly, the consumed energy is roughly 91 mAh per hour, drawing an average current of about 91 milliampere. Comparing these two test results, the energy saved by the use of the architecture is over 36 percent. However, the crucial detail with these results is that they have been obtained with 3G disabled.

5.4.2 3G

When looking at the measurements with 3G enabled (Figure 5.4), it is clearly visible how the energy consumption of the architecture grows even higher than with continuous use of the accuracy setting "Best".

As it has been the case in the preliminary tests of Chapter 3, the measurements with 3G active are more promising on the iPhone 5. Figure 5.5 shows, that the energy cost is larger in general for both measurements. Nevertheless, the average drawn current here is over 115 mA when using the desired accuracy "Best" and only 75 mA with the use of the architecture. The energy saved when comparing these two measurements is 35 percent.

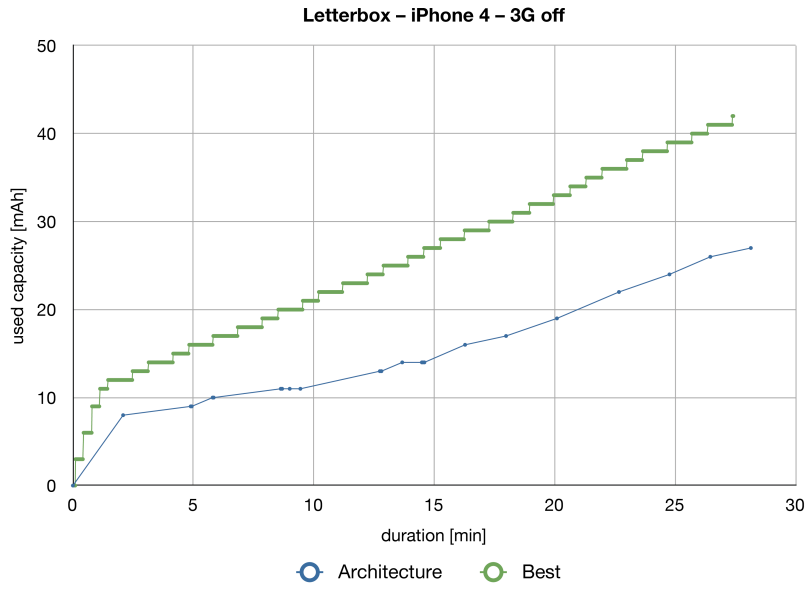


Figure 5.3: Used capacity on the iPhone 4. First, using the proposed architecture, then with accuracy set to "Best". 3G is disabled at both measurements.

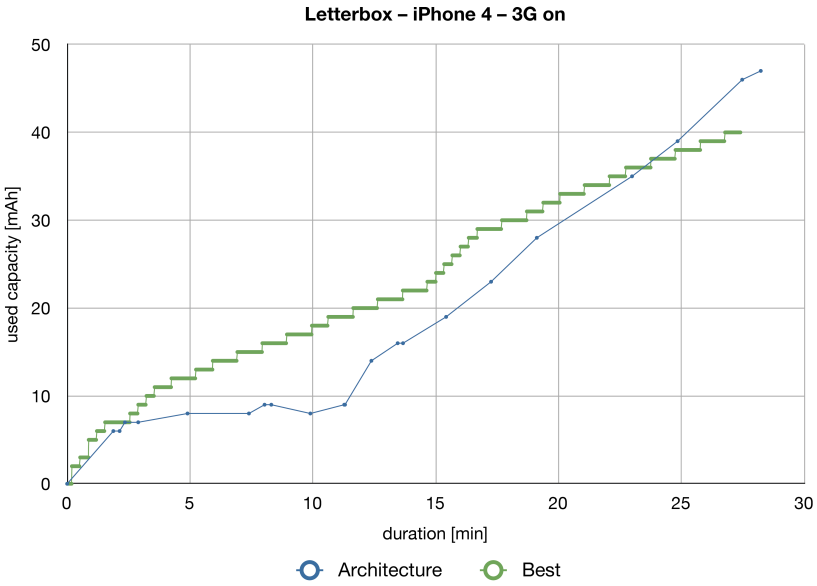


Figure 5.4: Used capacity over the duration of the test on the iPhone 4. First, using the proposed architecture, then with accuracy set to "Best". 3G is enabled at both measurements.

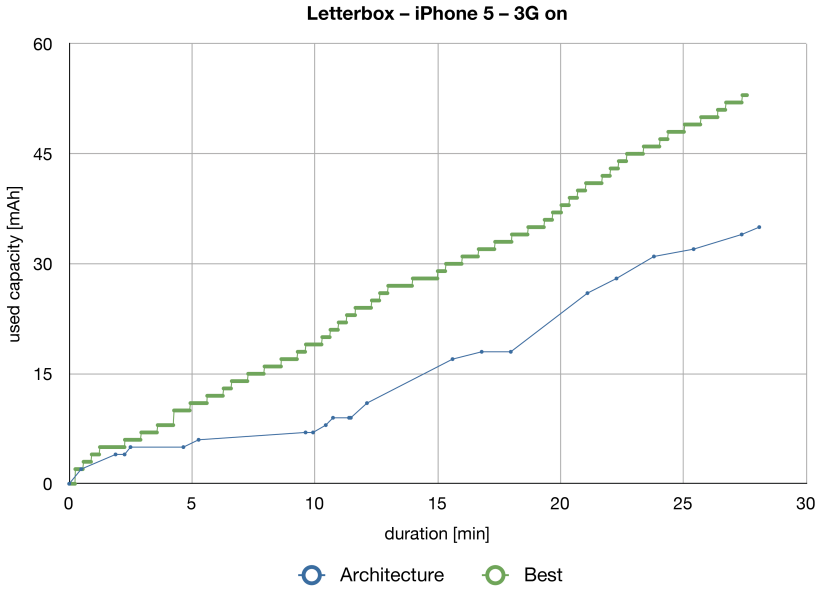


Figure 5.5: Used capacity over the duration of the test on the iPhone 5. First, using the proposed architecture, then with accuracy set to "Best". 3G is enabled for both measurements.

5.4.3 Accuracy

The percentages of energy saved with the use of the architecture on the iPhone 4—with 3G disabled—and iPhone 5 are considerably high, however the goal was also to not sacrifice accuracy at the user-relevant locations. The results reveal that this goal was not met. Figure 5.6 (iPhone 4) and 5.7 (iPhone 5) show that the actual accuracy at the point of the relevant locations is not as good as expected. The architecture registers when getting in the vicinity of the relevant locations and the accuracy increases to around 65 metres uncertainty, however it fails to register when getting closer than 100 metres to them.

Desired Accuracy Level

A comparison of the desired accuracy level and the actual accuracy during the experiment shows, that the architecture actually does go to the highest desired accuracy level some time before passing the relevant location, however does not remain there long enough to affect the actual accuracy (Figure 5.8).

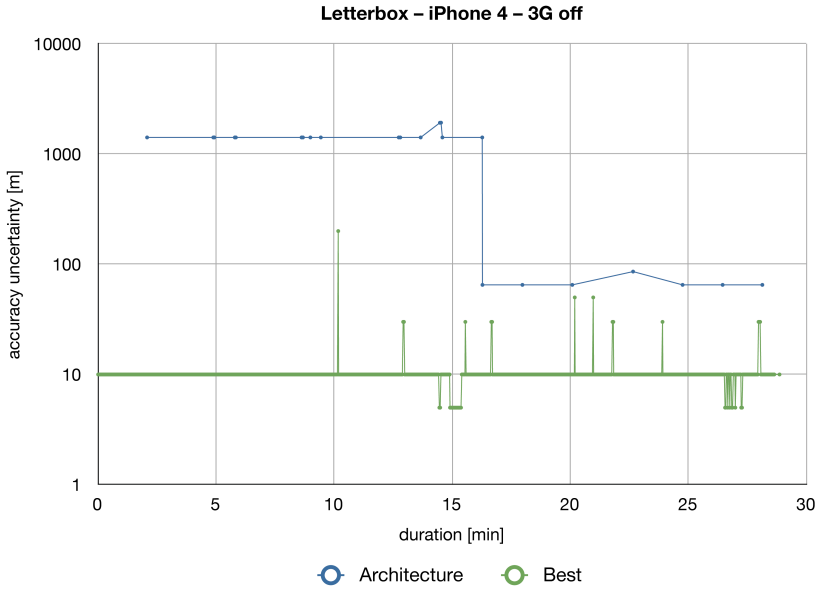


Figure 5.6: Accuracy uncertainty over duration of the tests on the iPhone 4 with `distanceFilter`-property set to one third of the distance to the nearest relevant location. The relevant location is passed at 19 minutes duration.

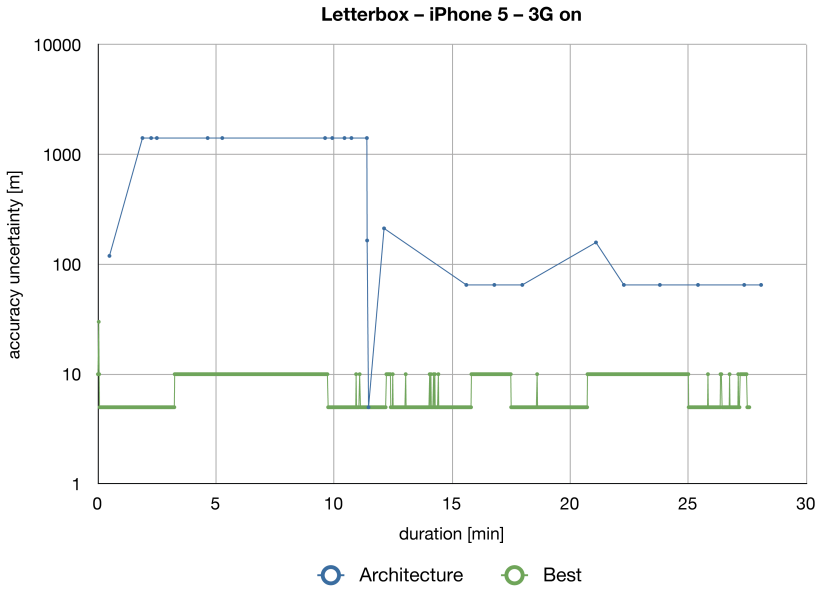


Figure 5.7: Accuracy uncertainty over duration of the tests on the iPhone 5 with `distanceFilter`-property set to one third of the distance to the nearest relevant location. The relevant location is passed at 19 minutes duration.

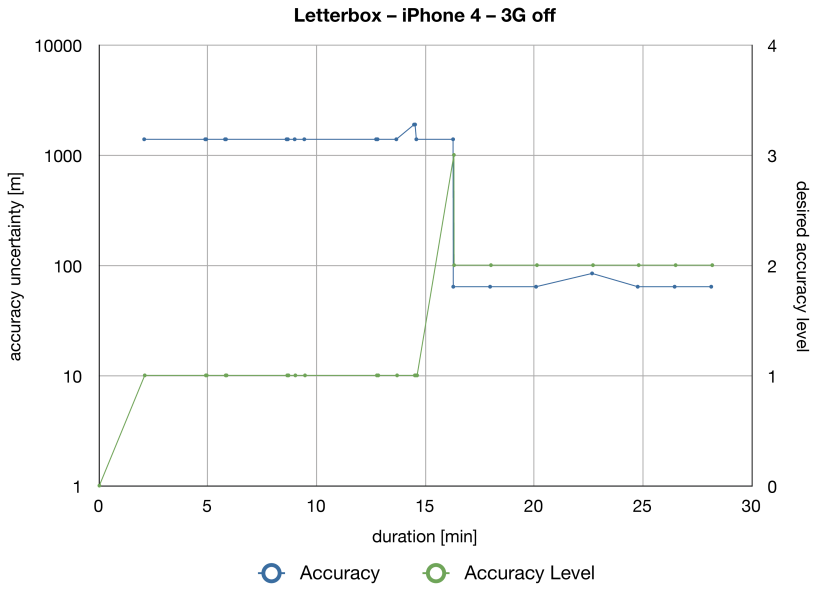


Figure 5.8: Comparison of the current desired accuracy level and the actual accuracy. The accuracy levels in this graph are 1 for "3 km", 2 for "100 m" and 3 for "Best" desired accuracy.

5.4.4 Variations

The actual accuracy of the results is not as proficient as expected. Two things might be responsible for this, which are either too high user velocity or flaws in the architecture. Due to the velocity of about 15 kilometres per hour during the tests, it might occur that the phone passes over the relevant location too quickly for the architecture to register. As the architecture in *Letterbox* is mainly used by pedestrians, new tests with less velocity are taken. Another possibility for the resulting accuracy faults could be due to flaws in the architecture itself.

Therefore another test has been conducted at pedestrian velocity with the architecture now leaning more towards favouring accuracy over energy-efficiency by changing the `checkDistancesToLocation`-method and omitting lines 42–45 of Program 4.2. This has proven to be the main flaw of the architecture. In doing so, the accuracy level will no longer be decreased based on the actual accuracy of the current location. This ensures earlier setting of higher accuracy at the cost of higher energy consumption. The results of the test at pedestrian velocity with the alteration of the architecture is presented in Figure 5.9. It shows that the accuracy changes from 65 metres to 10 metres shortly after 15 minutes, which was exactly when passing the relevant location.

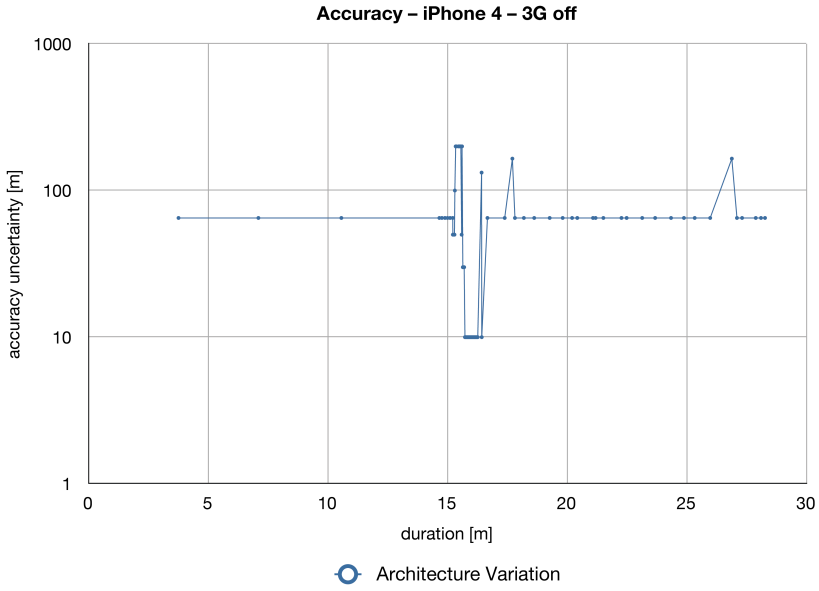


Figure 5.9: Accuracy uncertainty of the test at pedestrian velocity, with an alteration to the architecture.

Chapter 6

Evaluation

The results in the previous section show that the architecture has a significant impact on both energy-efficiency and accuracy of the application *Letterbox*. By conducting the preliminary tests, the effectiveness of the architecture could be evaluated more properly. Overall, conclusions can be drawn regarding energy-efficiency, accuracy and the practical value of the proposed architecture.

6.1 Energy-Efficiency

The main goal of the architecture is to maximise energy-efficiency when using the localisation systems available on the mobile device. The preliminary tests of energy cost and the results of the experiment reveal the main influences concerning energy-efficiency:

- device settings,
- network connection,
- desired accuracy.

6.1.1 Device Settings

The preliminary tests show, that a profound difference in energy consumption is made by toggling 3G in the cellular settings. This is highly noticeable in urban areas, where energy measurements return confusingly unexpected results when 3G is active, as shown in Figure 3.5. The assumption, that the dynamic selection of desired accuracy will have a proportional effect on energy cost is not supported by these results on the iPhone 4. Only when 3G is deactivated, the differences in energy cost are visible (Figure 3.8). Contrary to this, the 3G setting on the iPhone 5 does not have as much of

an influence on the measurements and the differences are noticeable with 3G active as well (Figure 3.10).

A substantial difference can also be seen when comparing the energy cost of the architecture with 3G once disabled (Figure 5.3) and once enabled (Figure 5.4). The iPhone 4 draws an average current of close to 100 milliampere with 3G enabled, but only slightly over 57 milliampere when it is disabled. This means that the device settings, which the architecture has no control over, have a very high influence on the energy-efficiency, without significantly affecting the location accuracy (Figure 6.1).

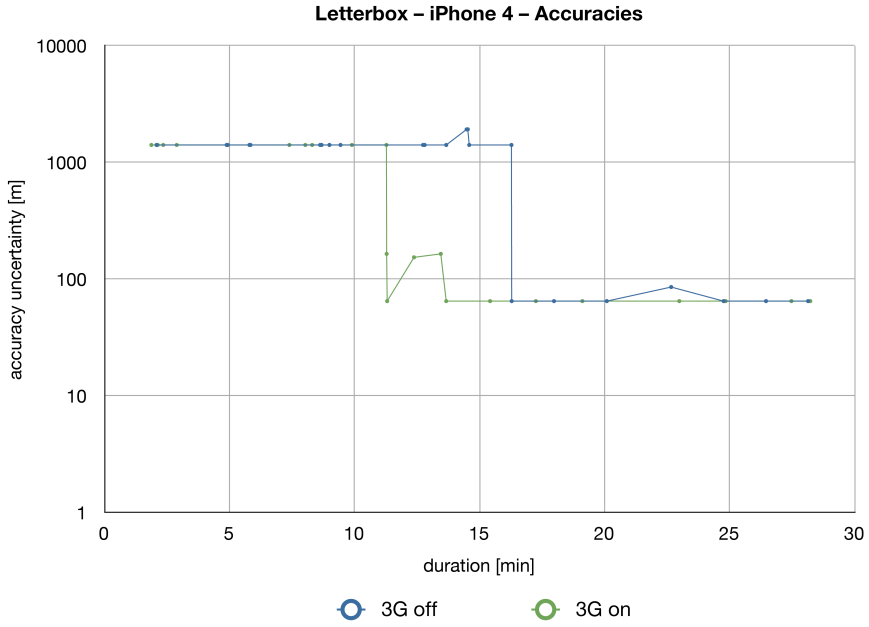


Figure 6.1: Comparison of the actual accuracy of the architecture with 3G once enabled, once disabled.

6.1.2 Network Connection

In order for the architecture to be energy-efficient, the assumption that the energy cost of the necessary network connection is smaller than the energy gain through the use of the architecture needs to be correct. The preliminary tests show, that the main factor here is the length of the update interval. As Figure 3.4 illustrates, the interval of 5 minutes, as compared to 1 minute, already saves more than half the energy needed. The chosen interval of 10 minutes for the architecture meets the expectations and the results of the experiment show, that the app is still using less energy with the architecture than without it.

However, there is one disadvantage with the increase of the update interval, which is the delay of the arrival of new content. For *Letterbox*, this means that a message could have a potential delay of 10 minutes until it arrives at the receiving user. This constitutes a problem for a messaging service such as *Letterbox*. For other applications, however, it might be tolerable.

6.1.3 Desired Accuracy

The preliminary tests confirm, that the energy consumption is, for the most part, proportional to the desired accuracy setting of the `CLLocationManager` of the *Core Location* framework, except on the iPhone 4 when 3G is activated in urban areas. In particular—due to the noticeable gap between the accuracy steps “100 m" and “10 m” (Figures 3.11 and 3.8)—the use of the lower accuracy setting is preferable, whenever possible.

6.2 Location Accuracy

Another major goal of the architecture is to keep high accuracy at user-relevant locations. In Chapter 3, the tests reveal that the actual location accuracy not always meets the desired accuracy setting. In the city, the accuracies mostly correspond to the desired level (Figure 3.9). In rural areas, however, the desired accuracies of “1 km” as well as “100 m” do not correspond to the actual accuracy values. This indicates the importance for the architecture to take the actual accuracy into account.

The experiment showed that the accuracy goal of the architecture, as it was described in Section 4.5, was not met. The fact that the tests were conducted at 15 kilometres per hour and due to design flaws in the architecture, this led to the result that the relevant locations were not registered. Through changes to the architecture—leaning more towards accuracy than energy-efficiency (Section 5.4.4)—and by conducting the test at pedestrian velocity, the accuracy goals were achieved.

6.3 Practical Value

All energy-efficient localisation systems introduced in Section 2.4 have a practical value for their respective target applications. *EnLoc* [4] is a predictive system for approximate localisation through exploiting user behaviour. *EnTracked* [6] focuses on the local prediction of pedestrian movement. The system *a-Loc* [7] is a more general approach and focuses on the energy-accuracy trade-off for application needs.

The value of the proposed architecture lies in high accuracy at relevant locations does not suffer while energy is saved. It has the same idea of energy-accuracy trade-off as *a-Loc*, however depending on the desired accuracy set-

ting of the *Core Location* framework of the *iOS SDK*, rather than being able to work with the location sensors directly. However, by focusing on relevant location-dependent content the energy-efficiency can be improved, while the accuracy around the content remains high.

The architecture can be applied to any location-based system using a set of relevant location-dependent content. Considering the amount of location-based applications that have appeared in the last decade and how the location sensors affect the battery life of the phone, the architecture can be quite valuable. Nonetheless, there are limitations to it:

- number of relevant locations,
- user velocity,
- device settings.

Firstly, the solution is only energy-efficient, if the number of relevant locations is within limits. If the density of the content gets too high, the architecture is rendered ineffective. Also, if the user is moving too fast, the architecture might fail to register relevant locations. Moreover, as mentioned previously in this chapter, an active 3G network drains the battery more quickly and also reduces the effects of the architecture—more so on the iPhone 4 than the iPhone 5.

6.4 Application in Letterbox

For the *Letterbox* app, the proposed architecture is valuable in numerous ways. Only new messages are considered as relevant locations. This results in a long lifetime of the battery of the phone, when no new messages are in the user's vicinity. Even if there are messages nearby, the energy saved when using the architecture is still noticeable.

The issue of insufficient accuracy at the relevant locations is partly solved by adapting the architecture as described in Section 5.4.4. However the constraint of low user velocity is still an obstacle. If the user velocity is too high, the locations might fail to register with the architecture.

By the use of server requests to update the relevant locations, the architecture always knows which locations to track more accurately. The only drawback of the use of the architecture for *Letterbox* here is the delay which is caused by the length of the update interval. Further implementation is necessary to counter this problem. A push notification server, for instance, might help to gain real-time message delivery, however the extended use of the network will also have an effect on battery life.

Chapter 7

Conclusion

This thesis proposed an energy-efficient architecture for mobile location-based systems while preserving accurate positioning at relevant locations. The key aspect of the architecture is its use of user-relevant location-dependent content provided by a server to decide when accuracy is more important than saving energy.

By conducting a series of preliminary tests, some assumptions could be made about energy usage and accuracy. These include that energy-consumption of the iPhone is proportional to the desired accuracy settings and that the energy cost of the network connection is smaller than the energy gain through the use of the architecture.

The architecture has proven to be energy-efficient in most situations. The experiment showed, that there are differences between the models of the phones, especially with regard to the 3G network setting. According to the results, 3G has a high impact on energy consumption and reduces the efficiency of the architecture, especially on the iPhone 4. The architecture is also limited concerning the velocity of the user, making it best suited for pedestrians. This is because at higher velocities, the accuracy at relevant locations decreases and the architecture might fail to register them completely.

In the case of *Letterbox*, the architecture demonstrates its value in increased battery lifetime. The application is expected to be used mainly at pedestrian velocity, keeping the accuracy problems contained. One drawback, however, is the delay of the content update caused by the architecture.

Future work in this area might include more extensive tests on the energy-accuracy trade-off of the architecture to further improve possible user velocity, accuracy and efficiency without sacrificing accuracy at user-relevant locations. In the case of *Letterbox*, a push-notification server could be implemented to help gain real-time message delivery.

Appendix A

Powergremlin

Main File PGPowerUtils.m

```
//
// Created by manuel on 11/9/12.
//
// To change the template use AppCode | Preferences | File Templates.
//


void PG_loadOSDBattery(void) {
    NSString *path = @"/System/Library/PrivateFrameworks/GAIA.framework";
    NSBundle *b = [NSBundle bundleWithPath:path];
    if (![b load]) {
        @throw [NSError errorWithDomain:@"com.palominolabs.UnableToLoadGAIA"
                                   code:0
                               userInfo:nil];
    }
}

id PG_getSKController() {
    Class SKHIDController = NSClassFromString(@"SKHIDController");
    return [SKHIDController performSelector:@selector(sharedInstance)];
}

float PG_getTemperature() {
    return [[PG_getSKController() valueForKey:@"temperature"] floatValue];
}

id PG_getOSDBattery(void) {
    Class OSDBattery = NSClassFromString(@"OSDBattery");
    return [OSDBattery performSelector:@selector(sharedInstance)];
}

int PG_getRawBatteryVoltage(void) {
    return [[PG_getOSDBattery() valueForKey:@"getRawBatteryVoltage"] intValue];
}
```

```
int PG_getBatteryLevel(void) {
    return [[PG_getOSDBattery() valueForKey:@"getBatteryLevel"] intValue];
}

int PG_getBatteryCurrentCapacity(void) {
    return [[PG_getOSDBattery() valueForKey:@"getBatteryCurrentCapacity"] intValue];
}

int PG_getBatteryDesignCapacity(void) {
    return [[PG_getOSDBattery() valueForKey:@"getBatteryDesignCapacity"] intValue];
}

int PG_getBatteryMaxCapacity(void) {
    return [[PG_getOSDBattery() valueForKey:@"getBatteryMaxCapacity"] intValue];
}

int PG_getChargerCurrent(void) {
    return [[PG_getOSDBattery() valueForKey:@"getChargerCurrent"] intValue];
}
```

Appendix B

CD-ROM Contents

Format: CD-ROM, Single Layer, ISO9660

B.1 Thesis

Path: /Thesis

thesis.pdf	Energy-Efficient Location-Based System Architecture for Mobile Client-Server Applications using Location-Dependent Content (full thesis)

B.2 Experiments

Path: /Project/EnergyTest/EnergyTest

EnergyTest/*	preliminary test app source
Results/Preliminary/* .	results of the preliminary tests
Results/Letterbox/* . .	results of the architecture tests

B.3 Letterbox Client

Path: /Project/Letterbox-Client/Letterbox

Classes/*	user interface classes
Libs/*	used libraries and tools (including *Powergremlin*)
Modules/*	`DBModule`, `LocationModule`, `ServerModule` and `TestModule`

B.4 Letterbox Server

Path: /Project/Letterbox-Server

symfony	symfony execution file
config/doctrine/* . . .	database definition
lib/model/doctrine/* .	database entity classes
plugins/ddAuthPlugin/*	user authentication plugin

Path: /Project/Letterbox-Server/apps/api

config/routing.yml . . .	API routing
modules/*	API modules and classes

B.5 Online Sources

Path: /Sources

AppleOnLocation.pdf .	press release of *Apple* on the collection of WiFi location data
MobileMajorityUS.pdf .	statistics on smartphone ownership in the U.S.
Powergremlin.pdf	blog entry introducing *Powergremlin*

References

[1] D.W. Allan et al. *The Science of Timekeeping*. Application note 1289. Hewlett-Packard, 1997.

[2] James J. Caffery and Gordon L. Stüber. "Overview of Radiolocation in CDMA Cellular Systems". In: *IEEE Communications Magazine* 36.4 (1998), pp. 38–45.

[3] Yongguang Chen and Hisashi Kobayashi. "Signal Strength Based Indoor Geolocation". In: *IEEE Internatoinal Conference on Communications*. (New York City). Vol. 0. 1. Apr. 2002, pp. 436–439.

[4] I. Constandache et al. "EnLoc: Energy-Efficient Localization for Mobile Phones". In: *IEEE INFOCOM 2009 - 28th IEEE International Conference on Computer Communications*. (Rio de Janeiro). Vol. 28. 1. Apr. 2009, pp. 2716–2720.

[5] James Cowling. "Dynamic Location Management in Heterogeneous Cellular Networks". BA thesis. Sydney: University of Sydney, School of Information Technologies, Advanced Networks Research Group, Oct. 2004.

[6] Mikkel Baun Kjaergaard et al. "EnTracked: energy-efficient robust position tracking for mobile devices". In: *Proceedings of the 7th international conference on Mobile systems, applications, and services*. (Wroclaw). MobiSys '09. New York, NY, USA: ACM, June 2009, pp. 221–234.

[7] Kaisen Lin et al. "Energy-accuracy trade-off for continuous mobile device location". In: *Proceedings of the 8th international conference on Mobile systems, applications, and services*. (San Francisco). MobiSys '10. New York, NY, USA: ACM, June 2010, pp. 285–298.

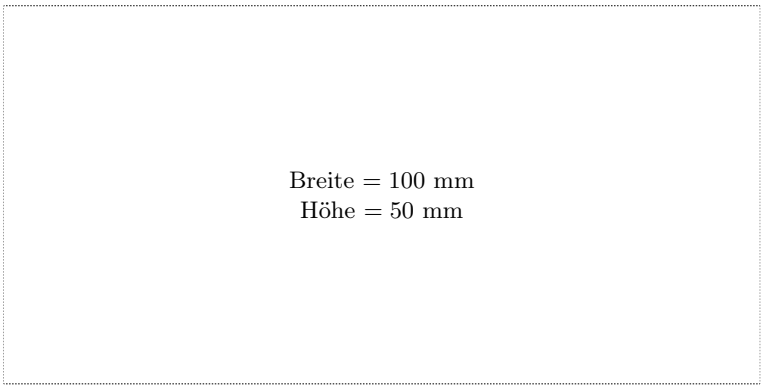
[8] Xiao Ma, Yong Cui, and Ivan Stojmenovic. "Energy Efficiency on Location Based Applications in Mobile Cloud Computing: A Survey". In: *Procedia Computer Science* 10 (2012), pp. 577–584.

[9] Jeongyeup Paek, Joongheon Kim, and Ramesh Govindan. "Energy-efficient rate-adaptive GPS-based positioning for smartphones". In: *Proceedings of the 8th international conference on Mobile systems, applications, and services*. (San Francisco). MobiSys '10. New York, NY, USA: ACM, June 2010, pp. 299–314.

[10] Arvind Thiagarajan. "Probabilistic Models For Mobile Phone Trajectory Estimation by". PhD thesis. Massachusetts: Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, Sept. 2011.

Messbox zur Druckkontrolle

— Druckgröße kontrollieren! —



— Diese Seite nach dem Druck entfernen! —