An adaptive approach to mobile sampling

Radu Călin Gătej

Supervisors:

Sune Lehmann
Jakob Eg Larsen
Piotr Sapieżyński

Kongens Lyngby 2013
IMM-M.Sc.-2013-80

Technical University of Denmark
Informatics and Mathematical Modelling
Building 321, DK-2800 Kongens Lyngby, Denmark
Phone +45 45253351, Fax +45 45882673
reception@imm.dtu.dk
www.imm.dtu.dk IMM-M.Sc.-2013-80

Summary (English)

The increase in the number of hardware sensors that smartphones come equipped with in the past several years has determined the emergence of a new area for research and application development, called mobile sensing. Mobile sensing applications make use of the different sensors that a smartphone packs, to determine as much as possible about the context of a smartphone user. Location can be determined from the GPS sensor, physical activity could be determined from the accelerometer, or the presence of other users, using Bluetooth. Along the multitude of other sensors built into a smartphone, information about the user and his daily habits can be analyzed in order to help improve, for example, the users health. By combining together data gathered this way from groups of people and even entire communities, behavioral patterns for specific groups of people or for humans in general can be studied.

The issue that mobile sensing applications face is the impact that they have on the battery of a smartphone. Due to sampling a high number of sensors at the same time, some of them more battery costly than others (Global Positioning System sensor), the time in which data can be collected while the user mobile is limited. To reduce battery depletion, mobile sensing applications usually make a compromise by reducing the temporal resolution of sensor sampling. This thesis proposes an improvement in both battery life and temporal resolution of the data by using adaptive sampling, a technique which conditions sensor sampling on the context of the user, rather then periodically sampling the sensors. The solution is designed, implemented and tested to work with the SensibleDTU project, which is a mobile sensing experiment that aims to study the mobility and personal interaction of students at the Technical University of Denmark.

The proposed solution intends to decrease the use of the GPS sensor for location estimation, by using a local database of WiFi access points that have been already mapped, as WiFi is a cheaper sensor than GPS battery-wise. GPS scanning would thus be conditioned by the presence of such access points. The mapping of WiFi access points is designed and implemented in this thesis by combining WiFi and location data gathered over four months from 160 students.

The adaptive sampling scheme proposed in this project also adds the mobility of the user as a condition for triggering the GPS, since location data is more useful if the user changes his position.

Finally, efficiency of the implemented adaptive sampling scheme is evaluated in terms of battery usage, temporal resolution and accuracy of data.

Preface

This thesis was prepared at the department of Informatics and Mathematical Modelling at the Technical University of Denmark in fulfillment of the requirements for acquiring an M.Sc. in Informatics.

The thesis deals with the improvement of energy efficiency and sensor sampling time resolution of a mobile application destined to be used in the mobile sensing experiment SensibleDTU.

The thesis consists of an introduction to mobile sensing and the SensibleDTU experiment (chapter 1), theoretical documentation and background on adaptive sampling (chapter 2), the design of an adaptive sampling architecture for the SensibleDTU experiment (chapter 3), the implementation of the adaptive sampling scheme (chapter 4), testing using test cases suitable for the SensibleDTU experiment (chapter 5), conclusions on the results of testing (chapter 6) and future work on adaptive sampling in the SensibleDTU experiment (chapter 7).

Lyngby, 22-07-2013

[illegible]

Radu Călin Gătej

Acknowledgements

First and foremost I would like to thank Piotr Sapieżyński for his endless support and patience in all phases of this project. Thank you for answering my countless questions, even at crazy hours in the night.

I would also like to thank Sune Lehmann for being a great thesis advisor and teacher. I am thankful for all of the support I've got from you during this whole project, and for being one of the coolest teachers I've ever had.

I'd like to thank both Sune and Jakob Eg Larsen for giving me the opportunity to work on one of the most interesting projects that I've encountered so far.

I'd like to thank the other people that helped me during this project, Jakob Eg Larsen, Vedran Sekara, Sebastien Thun.

I'd like to thank Piotr Sapieżyński, Vedran Sekara, Sebastien Thun and Konrad Stanek for creating a great work atmosphere during this project.

Many thanks go to Anamaria Pop, Sergi Rotger Griful and Marco Ottink for supporting me and proof reading my thesis.

I would also like to thank my workmates at Avaleo for creating a great work environment and supporting me throughout my Master period.

Last but not least, I would like to thank my parents for the incredible amount of support I've received from them during my whole Master period at DTU in Denmark.

Contents

CHAPTER 1

Introduction

1.1 Mobile sensing

Over the past several years, smartphones have developed an incredible amount of processing power and complexity, which has enabled them to become essential in daily life, as the main personal computing and communication devices. Along with improvements in computing resources such as memory or CPU, smartphones have been equipped with more and more sensors such as the GPS, the accelerometer, gyroscope, light sensors, which enables the devices to detect multiple parameters about the context of a user, such as location, physical activity level, geographical orientation. This myriad of sensors, combined with central role of the smartphone in everyday life, has created a new area for research and application development known as *mobile sensing*[LML+10]. This concept has found its way (very fast due to very accessible ways of developing and deploying of a mobile application) into many different areas of everyday life, such as healthcare[CMT+08], transportation[TRL+09] or social networking[CenceMe]. Sensors such as the accelerometer can be used to track a person's physical activity and draw conclusions on his day to day health[CMT+08], or the location sensor to provide crowd-sourced traffic information[TRL+09]. A very popular application created by Google, Google Now, [Inc13], uses the location sensor and social media information to show the user real-time traffic information about routes detected by the application as frequented by the user.

Nicholas et al. classify[LML+10] mobile sensing projects by their scale. *Personal mobile sensing* refers to projects that focus on monitoring analyzing just the individual, such as[CMT+08] which monitors a user's personal health and gives him feedback on how to improve it. *Group mobile sensing* refers to projects that gather data from a group of test subjects such as the students from an university[DTU12] or an entire community[RCK+10]. Results gathered from group sensing can be used to discover patterns and draw conclusions on general behavior of certain types of communities (students, city dwellers) or even on behavior patterns of humans in general[SQBB10].

1.2 SensibleDTU

SensibleDTU is an ongoing experiment at the Technical University of Denmark. The purpose of the experiment is to study the mobility and personal interaction patterns of students inside and outside the campus. In order to achieve this, smartphones have been handed out to 160 students, with a mobile sensing application running in the background which gathers different phone sensor data including accelerometer, Bluetooth, WiFi, GPS and others. The project has been running since October 2012 and in September 2013 it intends to expand the scale of the project to 1000 students, which should increase accuracy of the conclusions drawn on the studied behaviors.

The SensibleDTU project, and mobile sensing in general, are confronted with the issue of battery life. Having a set of sensor run frequently on the phone can be quite costly in terms of energy, especially when needing high temporal resolution of the data. If it also considered that students use the phones as their main communication devices, therefore running different other applications in addition to the experiment, then the battery life expectancy is not that high. In tests performed so far, the battery lasts on average about 12 to 14 hours (about a day). This is not sufficient in cases where students don't arrive home to charge their phones until late into the night (parties, late study sessions), cases which would also be interesting to study, since it can reveal information about student mobility. A higher temporal resolution of the location data is also needed, since the current interval for retrieving locations is not performant enough in cases where the user travels at a high speed or finds himself in places that cannot be localized accurately by the GPS, such as indoor environments. As will be explained in the upcoming section, both of these problems can be addressed using *adaptive sampling*, a technique that schedules sensor sampling based on events in the user context, rather then on periods of time.

Chapter 2

Adaptive sampling

One of the biggest challenge that mobile sensing applications face is the fast depletion of battery due to frequent sampling of more sensors. One way of reducing the impact on battery life is to simply increase the time interval in which a sensor is sampled based on how expensive energy-wise the respective sensor is. The result would therefore be a set of sensors which are sampled at a certain time interval based on their impact on batter life. This technique is known as sensor ***duty-cycling*** and it is the simplest way of performing mobile sensing. The disadvantage of this technique, however, is the fact that a tradeoff between resolution and energy depletion has to be done when establishing the sampling period. In the case of sensors that need a lot of energy, such as the GPS, the end result would be a time interval that provides neither the best resolution nor the best energy saving. As will be argued in the next section, a possible solution to this problem would be changing the sampling interval "on the fly" in order to better suit both issues.

2.1 What is adaptive sampling?

Adaptive sampling basically means dynamically changing the sampling interval of a sensor based on on certain triggering conditions. It can be classified in two types:

1. **Sampling triggered by other sensors**, where how frequent data is retrieved from one sensor, or even retrieved at all, depends on the output of another sensor (e.g. waiting for input from the accelerometer about movement before starting up the GPS).

2. **Sampling triggered by a sensor's own output**, where the sampling interval period is dynamically adjusted based on previously retrieved data of the same sensor. The sampling frequency increases or decreases based on how "useful" the previously recorded data has been(determined by a classifier), and the speed with which the interval is modified is usually based on certain function(linear, exponential, etc). As an example, the sampling frequency of the WiFi would decrease when a classifier would detect little or no change in the access points detected in previously recorded samples, and increase when the detected access points vary with each subsequent scan.

The benefits of adaptive sampling for battery saving consist in the fact that energy is only used when useful data is available. As an example of useful and not useful data, let there be the case of deciding how often to sample the GPS sensor, one of the most expensive sensors on a mobile phone. It would not make sense to continue to sample the GPS at the same high rate if, say, two or three consecutive estimations would show that the user is approximately in the same location. Similarly, when sampling the WiFi sensor, no detected access points or no change detected in the access points over the course of a few samples would mean that the sampling rate should go down, while more access points detected would mean an increase in the sampling rate. This way, energy can be spent on data that is really important, thus increasing the size of data that is useful for research. As proof of the effect of adaptive sampling on battery life, some previous experiments that made use of this method will be presented in the following section.

2.2 Related work

The various adaptive sampling techniques used in mobile sensing that will be presented have been used in experiments for determining the energy saving and as opposed to a basic form of duty cycling (even though all of them contain duty cycling in one way or another). The previous classification of adaptive sampling will be also used in grouping the existing methods.

2.2.1 Experiments using inter sensor sampling triggering

One of the most elaborate schemes of adaptive sampling employing this technique has been devised by the EEMS group[WLA+09]. Their method proposes sensor management by detecting specific states in which the user might be, states which can determine what kind of sensors need to be triggered. The detection of these states and the transition from one state to the other is determined through duty cycling of some specific sensors, which also draws energy. Energy efficiency comes from duty cycling lower power consuming sensors such as the accelerometer, and using them as triggers for sensors that use more of the battery such as the GPS or the microphone. They compare their results with basic duty cycling of the sensors and with another mobile sensing application, CenceMe[MLF+08], which are duty-cycling the sensors with respect to balance between accuracy and battery efficiency, and a simple periodical run of the sensors. The results, portrayed in figure 2.1., show that the inter sensor adaptive scheme of the EEMSS application comes out on top of the other two options in terms of battery efficiency.

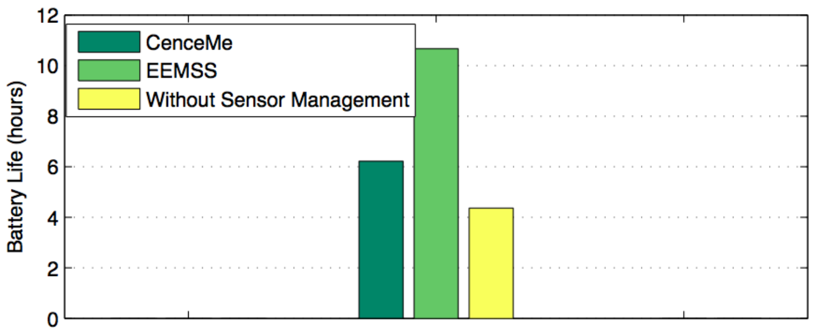


Figure 2.1: Comparison between EEMSS adaptive scheme, CenceMe adaptive scheme, and simple periodical sampling of the sensors[WLA+09]

2.2.2 Experiments using own output sensor triggering

In[RMM10] the authors perform a study on the effects of mobile sensing on battery life using as an adaptive sampling technique each probe's own output. They divide data coming from the probe in two types, "MISSABLE" and "UNMISSABLE" while using classifiers to detect each of these two states, which determine if the sampling frequency gets increased or decreased until certain defined thresholds. The paper mainly focuses on comparing the accuracy vs. energy consumed of different functions by which the frequency gets changed, such as linear, quadratic, exponential, or combinations of these. A dynamic adaptive sampling scheme is also used, meaning that if more unmissable events occur, the scheme switches to a more aggressive function. Their results show that these functions should be chosen based on the energy consumption of each sensors, as not all sensors behave in the same way when using the same function. In[Rac11], the authors experiment with the same type of adaptive sampling, also proposing a dynamic function that changes based on the context (i.e. number of unmissable events), which they demonstrate more efficient than in[RMM10]

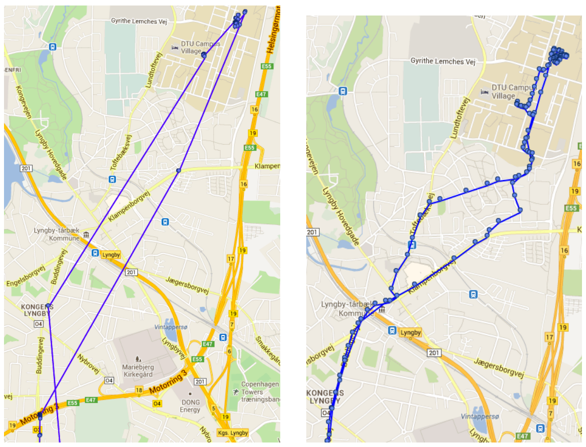
These examples of adaptive sampling techniques refer to experiments that have only used a smartphone as means of harvesting data. Other setups such as[citation needed], make use of special hardware installed in buildings where the experiments are conducted, or require the user to wear sensors in different parts of the body. In the SensibleDTU setup, just a smartphone is used, so that users are not disturbed from their daily activity. Previous work on this also shows that adaptive sampling can be a solution for saving battery in a mobile sensing environment, and an inter sensor sampling scheme design for the SensibleDTU application will be presented in the following section.

Chapter 3

Adaptive sampling in the SensibleDTU environment

As stated in the introduction, **SensibleDTU**'s goals as a mobile sensing experiment are to study patterns in students' mobility and their interaction with other students. In order to support these goals, the mobile application used for the experiment needs to provide **accurate** and **high resolution** location data for accurately tracking mobility, and to impact the **battery life** as little as possible, so that monitoring can cover situations when students stay out late into the night to interact with other students (prolonged studying sessions, parties, etc), which could mean a span of 20 hrs battery life.

The current iteration of the SensibleDTU mobile application samples 11 types of sensors, including the GPS, WiFi, and Bluetooth. Sensors that currently provide location information are GPS and the WiFi system, with the former one providing satellite locations, while the latter one is employed by the phone's operating system to retrieve locations from Google's database of WiFi access points with a known location. In the current run configuration, a location is provided every 15 minutes given the fact that the user is located either in a location with a clear view of the sky, or near an access points for which Google can provide a location. If the user of the phone voluntarily uses another location application, those locations are also registered. The average battery lifetime with this configuration is of about 12 - 14 hrs. A configuration of getting locations every 5 minutes was also experimented, but the battery life was shortened to 8 hrs. The first problem of this configuration is the temporal resolution of the location data. It can prove to be too low, especially in an environment where students frequently use bikes as means of transportation, leading to strange "jumps" of locations such as in 3.1. The second problem is



(a) (b)

Figure 3.1: Location estimation uncertainties ("jumps) due to low temporal resolution(a) compared to ground truth (b)

the battery life, which would be suitable in a typical use case of 9 AM to 11 PM. However, during weekends for example, students can be out and active as long as 4 or 5 AM, causing the experiment to miss quite interesting data related to student interaction outside of normal class hours.

3.1 Proposed adaptive sampling scheme

In order to address these issues, an adaptive sampling scheme based mainly on the WiFi sensor is proposed instead of the current configuration. The motivation for this is that information about the location of the user can be inferred from the data provided by the WiFi sensor. It is already known that Google, the developer of the operating system used in this experiment, maintains a database of WiFi access points across the world and their approximate location[Hou11]. In order to access these locations, the operating system makes use of the Internet. Instead of this, however, maintaining a database of trusted access points stored on the phone itself is proposed, thus eliminating the need of data transfer over the Internet, and the energy cost that comes with it. The experiment is mainly concerned with locations within the Copenhagen area, therefore the size of the database stored on the phone would not be a concern. The application would thus try to access locations provided by Google or the GPS satellites only when detected access points are not found in the local database. Accessing GPS is has the greatest impact on battery life from the sampled sensor in SensibleDTU application, and this solution would reduce the need to access this sensor (or access the Internet for the Google database) to only when there is no other option. Moreover, we propose that the configuration first tries to get a location from Google's database of WiFi routers, and only if this doesn't provide an accurate location should the GPS be scanned, being, as mentioned, the heaviest sensor on the battery and also the slowest in providing a location. WiFi is scanned in the current configuration every 10 minutes, but the operating system can increase the sampling frequency up until every 20 seconds. This solution should be significantly better as a location provider in terms of battery, since the system already schedules WiFi scans, and thus, additional strain on the battery is avoided. It would also be better in terms of location data resolution since it is sampled more often than the GPS in the current configuration. Last but not least, we propose introducing into the adaptive sampling scheme the condition of mobility, meaning that in order to scan for a location, the user has to have moved from his current location.

The local database of locations can be constructed from WiFi and location data previously gathered since the start of the experiment (October 2012). In the following section, existing methods for doing this will be presented, as well as the design of a solution that suits the current Sensible DTU environment.

3.2 Adaptive sampling using WiFi

In the mobile world, battery life has always been a big concern. At the same time, mobile phones' ability to provide location information of the user has been one if its greatest features. The classic way of providing location information has been through the use of the GPS, which despite its accuracy, has proven time and time again costly in terms of battery life. Because of this, mobile phone producers have turned to other ways of providing location through the mobile phone. The most common alternative to the GPS is the WiFi. As an example, Google uses WiFi gathered data from Android users, and through its Google Street View program, to map WiFi routers to locations. This enables to maintain a database of mapped routers which gets queried by mobile phones whenever they are near WiFi routers. This method, although providing locations with an accuracy of tens of meters (compared to the GPS which provides locations with accuracies of a couple of meters), is friendlier to the battery and faster, and has proven to be a performant alternative to GPS in situations where high location accuracy is not a requirement. Not to mention that it can provide locations in buildings and urban environments where a clear view of the sky might not be always available.

In the next subsection, a few existing methods of determining location using WiFi will be presented.

3.2.1 Related work

R. Henniges presents location estimation using WiFi, in his paper [Hen12], using four methods:

1. **Proximity sensing** This method simply assumes that if the user detects a WiFi access point which has a known position, the user is located in that position. In the case of more detected access points, the location of the one that has the strongest signal strength is selected. Accuracy can vary here, depending on the obstacles between the router and the mobile receiver, which can affect the range of the router. It can be used as a positioning method due to the short range of WiFi systems (tens of meters), which gets even shorter if there are walls or other obstacles between the access point and mobile phone.

2. **RSSI based trilateration** This method requires at least three access points to work. It uses the RSSI (Received Signal Strength Indicatior) to determine the distance to mobile device, and then, considering this distance as the radius of a circle, all three access points' radiuses are used to estimate the position (see Figure 3.2). This method requires that there are no obstacles between since they can greatly affect the signal strength based distance estimation.

3. **Triangulation method** This method requires at least two detected WiFi routers, and it determines the position of the user using the angle at which the radio weaves reach the mobile phone. This method however, requires modification of the router's hardware since it requires directional antennas to be able to determine the angle at which the signal hits the antenna.

4. **Location fingerprinting** This method proposes a slightly different approach. Instead of knowing the positions of the access points before hand, the signal strength of different routers at certain positions is recorded. This way, a radio map with vectors of routers and their respective signals at certain positions is constructed in the training phase, and then estimation of the location is done by computing the distance (n.b. Euclidean

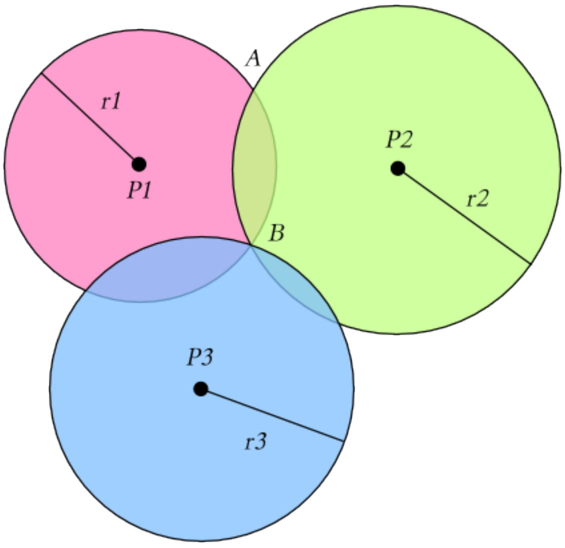


Figure 3.2: Trilateration(source: Wikipedia)

distance or any other form of plane distance) from a detected vector of routers and signal strength levels to the ones in the radio map, and the position of the closest vector in radio map is assumed. This method provides very good accuracy given that the initial radio map is accurately constructed[H^+08]. This means that signal strengths must be measured with a very fine granularity[H^+08].

As it will be seen in the following section, the resources available for the SensibleDTU experiment require a slightly different approach to the issue than the techniques previously presented.

3.2.2 Resources available

At the time of writing, the SensibleDTU experiment had run for 4 full months. Data gathered during this time can be used for building a map of access with an estimated location for each, so that the location of a mobile unit can be estimated with respect to the access points that it detects. The first thing that comes to mind when thinking of such a map is the radio map model used for location fingerprinting. However, as [Hen12] points out, it is quite tedious to construct a radio map that can be used for location fingerprinting due high location resolution requirements. In [H^+08], the author had to retrieve RSS information from a tested office (Figure 3.3) building floor in at least 60 places in order to construct a radio map, using 20 access points. Also, in order to have a radio map that can provide performant results, meaning a variation of error between 6 and 12 m in space as in Figure 3.2, the calibration had to be done in each calibration point with the mobile unit oriented in four directions. The data

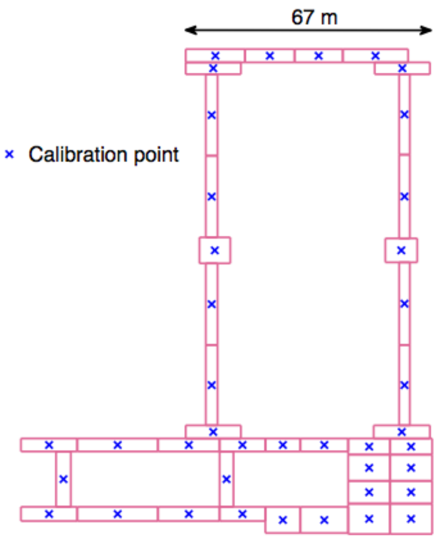


Figure 3.3: Division of an office space for creating a radio map[H^+08]

available in the 4 months comes from students with very different patterns of mobility. WiFi scans and location scans are not closely synchronized in time so as to say that a scan corresponds accurately to a location. Due to the fact that users have different degrees of mobility, the density of calibration points can vary, given the size of the area in which the experiment takes place (Copenhagen area). Also, there was no systematic retrieval of the RSSI in different orientations. It was concluded that it is quite difficult to obtain a radio map that can accurately be used for location estimation in the SensibleDTU environment.

Below, a different approach to building a map of access points with estimated locations is presented. The data that will be used consists of **456.161** recorded unique access points, **2.927.355** locations, out of which locations that have a reported accuracy of **30 m** (representing the radius in which the real location might be) or less will be used. Locations with such accuracy constitute 73 percent of the location data. As a comparison, The dataset contains about 18 million WiFi scans , retrieved from 162 users over the course of the months October, November and December 2012, and January 2013.

3.2.3 Proposed strategy

As mentioned previously, the construction of a map of access points with an estimated location is proposed. This would allow avoiding the sampling of the GPS sensor for locations if one of the trusted access points is encountered. The process of building this map would essentially consist of finding a set of possible locations for each access point, and finding a centroid of the set as the estimated location of the access point. The following steps describe how this process would look like in more detail:

1. *Finding a set of possible locations for each access point in the dataset*

   The approach to be taken here is to find all the location scans taken at the same time as WiFi scans, and for each access point detected in that WiFi scan, add the location to its list of possible locations, as well as signal strength of the WiFi scan. As it is quite unlikely that WiFi scans and location scans happen exactly at the same time, since the scans are not synchronized, a tolerance interval between the location scan and the WiFi scan should be used. Special care should be taken of the fact that the mobility degree of the user is not known, and therefore, during the selected interval between a WiFi scan and a location scan, the user might move very little, keeping the WiFi scan quite close to the actual location reported by the scan, or he might be in a vehicle, thus making the distance between a location scan and a WiFi scan long in space, even if short in time.

2. *Using the centroid of the location set as an estimate for the actual location of the access point*

   Having collected a list of possible locations for each access point, an estimation of its location can now be made. The mean of the locations should be good for finding a central location, however other methods should be experimented, such as the weighted mean based on the signal strength, or the median. The degree of trust in the estimation should be assessed by computing the average deviation of all the possible locations from the central estimated location. This basically means that the more that access point has been observed in roughly the same place, the more likely it is to actually be there. The time frame on which the possible locations of the access point spread should be experimented with, since a bigger time frame of observation might give more accurate results. Special care should be taken here to mobile hotspots (mobile phones turned into WiFi access points, or public transportation WiFis) since their location changes often over time. This problem can be solved partially by eliminating access points that have a network name which indicates a mobile hotspot (e.g. "Android hotspot") and by making sure that the locations have happened in a large time frame (the more days the access point has been observed in the same location, the more likely it is to be fixed in that location). Thresholds for the minimum number of possible locations should be experimented, as well as for the deviation.

3.2.4 Expected outcome

The expected result of this strategy is a map of access points with estimated locations, which have a low deviation (as explained above) and are very likely to be fixed. It is expected that most of the access points will be concentrated around the DTU campus given that the users of the application are DTU students, and the experiment had run during a semester. The usefulness of the generated map will be evaluated using two criteria:

1. **The prevalence of the mapped access points in the existing dataset**

   This means that the access points for which a location has been estimated should appear as often as possible in the scans, since this would mean that users often pass through those places.

2. **The accuracy of the estimated locations compared to ground truth**

   This should be assessed by comparing a subset of the map with access points for which ground truth is available. Currently, there is access to the approximate locations of the access points in the DTU campus (information about the building number in which the access point is located). Field testing should also be performed to asses accuracy.

3.3 Adaptive sampling using mobility detection

As previously described, another component in the adaptive sampling scheme is conditioning location scanning on the activity level of the user. In short, this means that before scanning for a location, the system should first check if the user has moved away significantly from his previous already known location. This is desired in the cases in which users are at their homes for a long time (e.g. sleeping) or when they are in a 2 hrs lecture for example. In those cases, the user does not move for a longer period than the duty cycle period and it would be redundant to scan for locations more often than that. This is not a new idea, the [WLA$^+$09] group also implements this making use of the accelerometer, which can be used for detecting different types of activities such as walking, riding in a car, on a bike, etc. However, this project proposes making use of the WiFi as a means of detecting if a user has moved. This method is described in [Sap13] and it demonstrates that a change in the strongest WiFi router indicates that the user has moved from his current location. This method should be preferred to the accelerometer method since it would make the implementation much simpler by not involving an extra sensor. Also, the movement detection heuristic is quite simple to implement and it would therefore not cause too much load on the processor, thus not affecting the battery.

The mobility detection component should pose the first condition after a WiFi scan has been performed: if the detected access points indicate that the user has moved, then the system should carry on with first looking in the local database of access points, and then if none if the detected access points is in the database, it should continue with scanning for a location from the Google database of routers, and finally from the satellites.

CHAPTER 4

Implementation

The target of battery improvement using adaptive sampling is, as mentioned before, the mobile application of the mobile sensing SensibleDTU experiment. The application is based on an open source mobile sensing framework called **Funf**[API$^+$11], which runs on the Android mobile operating system. It is meant to aid researchers in harvesting data in a secure and reliable way, while keeping the impact on the user that is using the phone, as low as possible. Since the implementation of the adaptive sampling techniques presented will involve extending the Funf code base, a high level description of its architecture follows.

4.1 Adaptive sampling using Funf

The central concept in Funf is the probe. Probes are background Android services[Goo13b] which are scheduled to run at a specified time interval for a specified duration of time. The framework contains probes that sample many of the phone's sensors (location, bluetooth, accelerometer, etc). Configuration of the mobile sensing scheme is done through a JSON file where the probes that are supposed to run are specified, along with the interval and duration of the run. The application runs in the background of the phone, so as to not disturb the user experience of the phone's user. After the probe samples the sensor at the specified interval, the the retrieved data is stored in a database, to which other applications have no access. At a pre-specified interval, the database is encrypted and dumped to a file on the phone's external storage[1]. Finally, when the phone connects to a WiFi Internet connection, the stored data file is sent to a central server to be analyzed by researchers.

In order to implement the adaptive sampling scheme proposed in Chapter 3, the Funf framework is extended with a new probe, called *WifiLocation probe*, which has the purpose of analyzing the data captured by the WiFi probe and decide if the location[Fun13b] probe should be turned on. The WifiLocation probe is no longer scheduled by the Funf framework to run, but instead uses the *Delegate probe scheduler*[Fun13a], which delegates the triggering responsibility to a chosen probe, in this case the WiFi probe. The WifiLocation probe is thus triggered whenever the WiFi probe runs, and has access to the data retrieved by the WiFi probe.

The first step in the adaptive sampling scheme is evaluating whether the user has moved or not (algorithm explained in more detail in section 4.3), and afterwards, if the algorithm indicates movement, the detected access points are compared with a list of access points that have a location estimate ("known"), as described in 3.2.4, in order to determine if the location probe should be run or not. The list of known access points is stored under the form of a flat file containing the MAC addresses[Wik13b] of the access points, and loaded into memory at the start of the application. If there is at one detected access points that is found in the list, the probe does nothing, as the WiFi data, which was already retrieved by the Wifi probe, will be analyzed in post processing on the server in order to extract the location. If there is no known access point detected, the location probe is fired[2], which accesses the GPS satellites and the Google access point database (using Android's LocationManager[Goo13a]), and stores the location that is the most accurate. The duty cycling interval of the location probe is maintained if there are no known access points, therefore the location probe is still constrained to run not more often than 15 minutes as in the previous configuration.

The WiFi probe is scheduled to run every 10 minutes, however it is programmed to "piggy-back" on other uses of the WiFi receiver, such as when Android looks for new WiFi routers. This means that the probe might run at intervals ranging from every 20 seconds to a maximum of 10 minutes as it is scheduled.

[1]Applications with root access to the phones can still gain access to the internal databases of other Android applications. Encrypting the Funf database should be considered in future versions of the framework

[2]Note that triggering runs of the location probe without having it scheduled through the initial configuration might affect the results, since in the implementation provided in Funf for triggering a probe at any time does not easily provide control over the run time of the probe

The location probe, on each run, requests locations from Android's network provider (which accesses Google's database of access points) and the satellites, at the same time. Battery could be improved here by asking the network provider first, as it is more battery friendly, and if the needed accuracy is not supplied, the GPS provider should be started.

The smartphones used for testing are Samsung Galaxy phones with the latest iteration of the Android operating system, Jelly Bean 4.2.2. For the purpose of testing different configuration on multiple phones at the same time, the batteries of the test phones were compared by running the same application on all of them, so as to establish their level compared to a phone selected as reference.

In the following section, the implementation steps taken for creating the map of trusted access points are explained.

4.2 WiFi map creation

The project databases contain meta-information about each data sample (including user id, timestamp, probe name) as well as the actual sample data. Here, I combine the output from the WiFi probe and the Location probe in form of JSON dumps of the database in order to estimate physical location of the discovered WiFi routers. As the data gathered in 4 months was numerous, due to the sampling interval of each of the two probes (10 minutes or less for the WiFi, and 15 minutes for the GPS), the size of each file was quite large, 22 GB for the WiFi file and 2 GB for the location file. This is inconvenient for data analysis, as parsing files of such size takes a large amount of time and resources. Therefore the huge file of the two, the one containing WiFi data, was split up into four SQLite3[SQL13] databases, corresponding to each of the four months, and the data that was stored in them contained only information that was actually needed in the analysis (BSSID, SSID, signal strength, user id and timestamp). Other information such as the security type or other technical information about the unit was discarded.

Analysis of the data was performed using Python as it is a convenient scripting language for fast data analysis. The implementation of the steps required to build the map follows, as it was proposed in 3.1.3.

4.2.1 Finding a set of possible locations for each access point

The **first** step in building the map of trusted access points is finding a set of possible locations for each one of them, which we will call *observations*. This part of the algorithm starts with matching WiFi scans with location scans *time-wise*. A location estimate is considered an *observation* of a WiFi scan if both the WiFi scan and the location scan belong to the same user, and if the distance *in time* between the scans is not greater than a certain time threshold. The explanation for this is that ideally, if a WiFi scan happened at exactly the same time as a location scan, then it can be assumed with certainty that the WiFi scan happened at that location. Since this is not possible in our configuration, due to the fact that scans from different probes are run asynchronously, a time interval between the scans can be allowed, assuming that the user has not moved from that location, or has moved very little.

After the algorithm matches one WiFi scan (or more, since there could be more scans in the time interval) with one location scan, it assigns that location to each router from that specific WiFi scan. Experiments have been done with matching a location scan with strictly one WiFi scan in the time interval, the one that is the closest in time, since this method can eliminate situations where the user is extremely mobile in the short time interval (e.g. in a car).

The outcome of this step is a map-like data structure, having as keys the MAC addresses of each access point that had at least one observation, and as values a list of tuples containing the latitude, longitude, user id, timestamp of the WiFi scan, timestamp of the location scan, and the signal strength of the router at which the respective location was matched. Experiments were also performed with having in the list of observation just one location from the matching time interval, as this takes care of the situation where the user is moving fast and receiving location updates faster than 2 minutes (even though the location scan time interval is set at 15 minutes, the location probe also receives updates from other application that use the GPS independently of the SensibleDTU application).

4.2.2 Finding a centroid of the set of locations to use an estimate for the location of an access point

The **second** step, in the compilation of a map of access points, is using the acquired list of locations to extract their centroid. The measures experimented were the mean and the median. The 1D median is usually computed by sorting an array of values and taking the middle value. Calculating the 2D median for the given coordinates required, however, the use of the geometric median[Wik13a]. The geometric median represents the point from which the sum of distances to all the other points in the list is minimum. This was computed using the Weiszfeld algorithm[Wik13a]. Since the latitude and longitude are in polar coordinates[Wik13d], and computing the mean and median requires Carthesian coordinates, the latitude and longitude were converted to their equivalent Carthesian coordinates (the ECEF projection[Wik13d]), using the Haversine formula[cod13]. After the median is computed, its Carthesian coordinates are converted back to polar coordinates. The outcome is a map like structure which has as keys the mac adresses of the access points and the centroid location as values. Worth mentioning here is that due to a bug in the Android location system, some locations were misplaced in the Cisco campus in Texas. Therefore these locations were filtered out when computing the median.

4.2.3 Degree of trust in the estimation given by the centroid

The obtained list of access points with estimated locations was analyzed in terms of its degree of trust, and it was filtered based on a number of attributes that can influence this. The degree of trust is given by the deviation of the observations from the computed centroid. This is based on the heuristic that "the more observations of the access point in the same location, the more probable it is that the access point is there". The deviation was computed using the standard deviation when using the mean as a measure for the centroid, and using the median absolute deviation[Wik13c] for the median estimated centroid. The maximum accepted deviation from the centroid considered "good enough" was 30 meters.

The effect of the number of possible locations on the centroid was studied by computing the variation of the deviation with respect to the number of locations. This analysis was performed on access points with a number of observations ranging from 100 to 1000. Their deviations were computed by using a gradually increasing number of observations (chosen randomly), and a threshold was chosen after the point where the deviation would start oscillating only within 30 meters. The average number of observations at which the deviation would vary within 30 meters or lower, from all the analyzed access points, was then computed and used as a filtering condition.

Some of the access points were considered "mobile" by using their SSID (the network name). A set of network names that indicate that an access point is actually a mobile hotspot was compiled using the default network names that some popular smartphones use when setting up a mobile hotspot (iPhone, Nexus series, HTC series). Also the names of the WiFi networks offered by the public transportation in Copenhagen were added to this set. These names are specified in the trains and busses ("Bedrebustur" for buses and "CommuteNet" for trains).

4.2.4 Validation of the strategy

The validation of the map of trusted access points was considered using the two criteria proposed in 3.1.3. The set of access points that resulted from applying the strategy described above and filtering was compared with the entire initial set of WiFi scans to see how many of the trusted access points appear in the scans (assuming that if at least one access point appears in a scan the user is in that location), in order to determine how much of the locations frequented by the user group are covered by the map of trusted access points. The distance in meters between a WiFi location estimate and a GPS location estimate, where both scans were triggered at the same time or within maximum 5 seconds, was used for evaluating the performance of the WiFi location estimation.

Ground truth data for the access points belonging to the DTU Campus System was available in the form of a list of MAC addresses of access points mapped to a building number. These MAC were compared with the list of trusted access points and were represented on a map representing the DTU campus and the numbered buildings.

4.3 Implementing mobility detection

Detecting the mobility of the user, so as to condition the location probe sampling on it, was implemented using data provided by the WiFi probe. The implementation of the algorithm decides that the user has changed his position by observing if the access point which holds the strongest RSSI from all access points in a scan has changed compared to the previous scan. If there are multiple access points that have the strongest RSSI, it is considered that the user has not moved from a scan to the next, if at least one of them reappears in the subsequent scan. This is implemented in the WiFi location probe by doing the intersection of the set of the strongest access points in a scan with the set of strongest access points in the previous scan. There are two aspects that were considered in this specific implementation (as opposed to the one in[Sap13]):

1. **Mobile hotspots and public transport WiFi networks** Mobile hotspots are not taken into consideration in mobility recognition since they from the range of the phone's WiFi thus falsely asserting that the user has moved, when actually the person that initiated the hotspot has moved. Also, WiFi networks belonging to the public transportation system are not taken into account in order to detect mobility when the user is on the bus/train. If these networks were considered, the application would falsely show that the user is not moving since it would always detect the same signal strength level for the bus/train network. It would also falsely detect mobility in the case when the user is very close to a station and a bus or train stops.

2. **Variation of the RSSI between two scans, even though the user has not moved** It was observed during some preliminary tests of the mobile recognition implementation that the variation in the signal strength can affect the order of the strongest access points. A particular test revealed that even though the phone was held very close to an access point (less than 1 meter), and moved only from one hand to another, that specific access point was not always the strongest, even though other access points were at least about 6 meters away and behind walls. It is for this reason that implementation assumes that two or more access points are equally the strongest if the signal strength difference between them is less than 12 dB. This particular threshold was determined by observing the variations in RSSI when holding a test phone close to (30 cm) a test WiFi access point and taking the average distance in dB whenever this access point would not have the strongest RSSI in the list due to fluctuations of RSSI and small movements of the phone which should be classified as movement (e.g. moving it 50 cm away).

CHAPTER 5

Results

This section presents the results of the experiments described in the previous section. First, the results of creating the map of trusted access points are presented, followed by the results of the entire adaptive sampling scheme.

5.1 The map of trusted access points

The end result of the described map creation process can be seen in Fig. 5.1. It contains around **17000** WiFi access points mapped with an accuracy of under 30 meters. The results of each step in the map creation process are presented in the following subsections, together with the discussions and motivations for the decisions taken at each step.

5.1.1 The list of location observations for each access point

The effect of the time interval on the accuracy of the data is influenced primarily by the fact that in that time interval, a user can be static or walking on foot, which means that location observation is more accurate, and similarly, by the fact that a user might be in a car and thus changing her position significantly

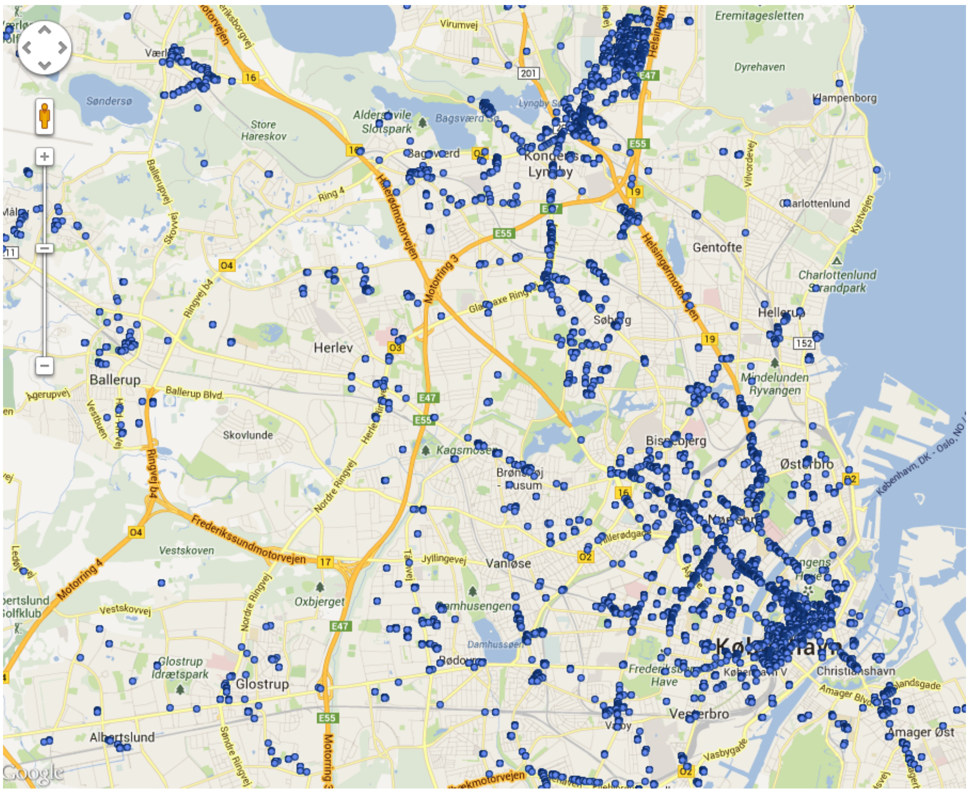


Figure 5.1: The map of over 15000 WiFi access points in the Copenhagen area (and other locations in Denmark not visible in this illustration)

in the chosen time window. However, the smaller the time interval, the less observations there are for each access point, and access points observed altogether (this is because WiFi and location scans are run independently from each other, and at different time intervals). A trade-off between the number of observations per access and the accuracy given by the small time interval is considered, thus ending up with the 2 minute interval.

The experiments done with selecting just the closest in time WiFi scan to a location scan in the chosen time interval, while also keeping just the closest location to a Wifi scan, with the purpose of diminishing the effects of mobile users has proven quite effective as can be seen in Figure 5.3. The problem is that since the Funf application also gets location data whenever other applications request it, when a user turns on a navigation app while he is in a car or on a bike, locations are updated very fast (every second) and therefore the same

WiFi scan would receive multiple locations (illustrated in Figure 5.2). Here, the blue dots represent the possible locations of an access point, i.e. the locations from which the user has observed an access point within the time interval of 2 minutes. From the timestamps of the locations that form the path, which are 1 second away from each other, it can be seen that there were more location scans in under 2 minutes.

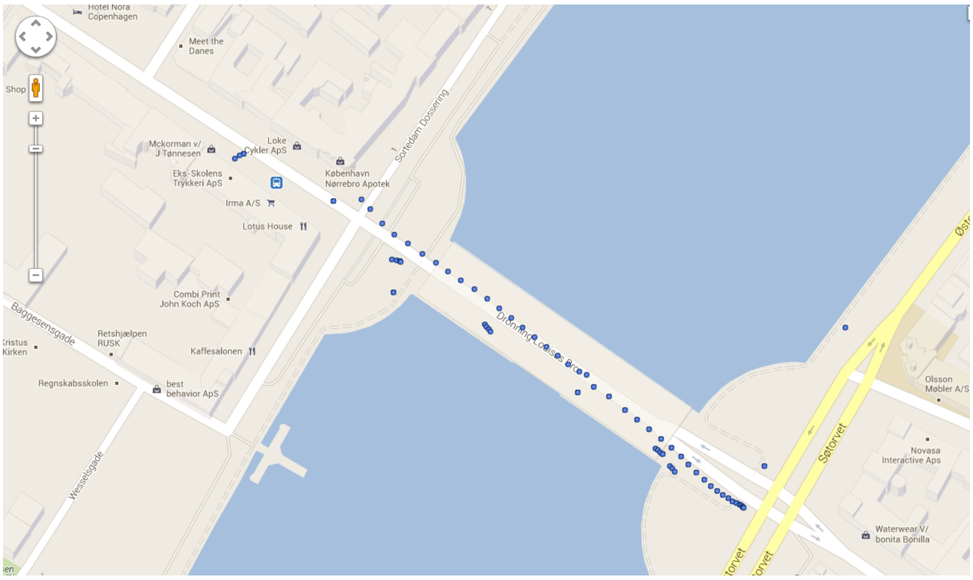


Figure 5.2: The location observations of one access point, with location updates more frequent than the time interval

After choosing just the closest location in time, the situation is improved as can be seen in Fig. 5.3.

As an effect of using this method, the average number of observations per access point has decreased to **136** and the number of access points that have observations has decreased to **56086**.

5.1.2 Estimating the location of an access point

In the second step of the map creation process, experiments were performed using the mean and median as a centroid estimation method. In figure 5.4, two

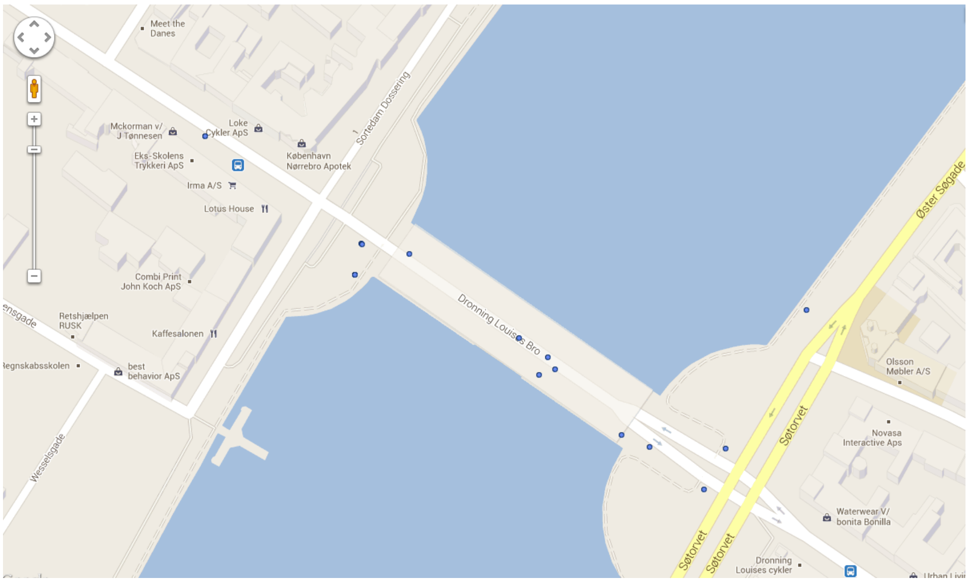


Figure 5.3: The location observations of one access point, when only the closest location scan to a WiFi scan in 2 minutes is chosen

of the location observations appear to be much further way from the cluster of observations (where the access point is most likely to be located due to the amount of observations in that area), probably due to the fact that the users were moving fast within the selected 2 minutes interval between a Wifi scan and a location scan. Their location on the highway indicates that they were probably in a car. The location estimation using the mean as centroid appears to be affected by the two outliers, since the mean centroid is placed at the edge of the cluster, towards the outliers. The median however, is positioned at the centre of the cluster, thus proving that it is more efficient against outliers than the mean. As a result, the median was selected as the favored measure for the location estimation of the access points.

5.1.3 Refining the map

The deviation of the access points from the centroid is, as stated in the implementation, the measure used for the accuracy of the location estimation. This was measured using both the standard deviation from the centroid, and the median absolute deviation from the centroid, which should be more effective

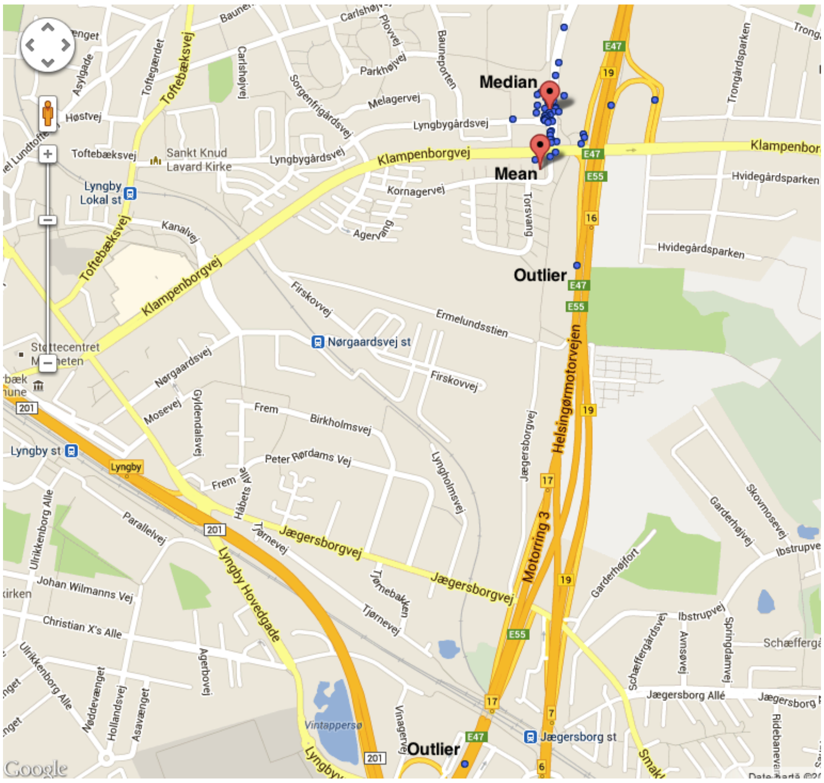


Figure 5.4: Location estimation using mean and median. The blue points represent the location observations, while the red ones represent centroids.

against outliers and report the deviation more accurately. The deviation histograms of the access points are illustrated when using the standard deviation (Figure 5.5) and the median absolute deviation (within 100 meters) (Figure 5.6). The one using the median deviation shows a lot more access points with a deviation under 30 meters than the one using standard deviation. The one using the median deviation shows a lot more access points with a deviation under 30 meters than the one using standard deviation.

The minimum number of location observations needed was thus set to **5** taking into consideration the number of access points that would be discarded using this threshold, and the fact that this number was sufficient for the deviation to stay within 30 meters.

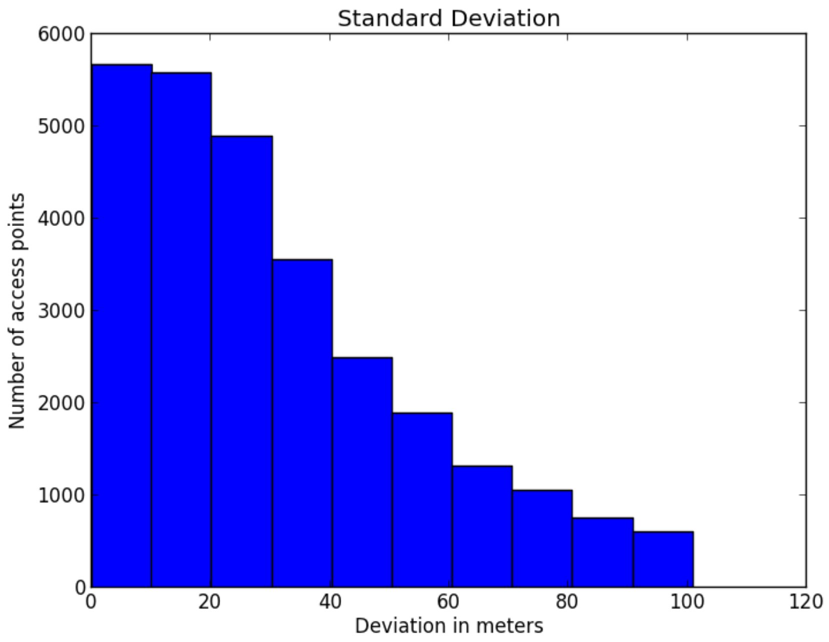


Figure 5.5: Histogram of access point deviations using the standard deviation

The number of WiFi access points which had an SSID corresponding to the public transportation WiFi system or default SSIDs of smartphones that can setup mobile hotspots resulted was reported at **6327**. These access points were removed from the set of trusted access points.

At this point, after filtering out location estimations that have a deviation greater than 30 meters, a number of observations smaller than 5 and have an SSID which indicates a mobile hotspot, the number of access points that can be used for estimating locations good enough for the SensibleDTU experiment has decreased to **17986**. It appears to be quite small compared to the total number of detected access points (the mapped access points represent 4 percent of the total detected access points), however, the usefulness of this subset of access points will be seen in the upcoming section, with regards to both the accuracy of the estimations compared to ground truth, and the coverage of the places visited by students. The geographical coverage of the final set of access points can be seen in Figure 5.7. The Copenhagen area is covered, with large concentrations of mapped access points in the DTU campus and central Copenhagen.

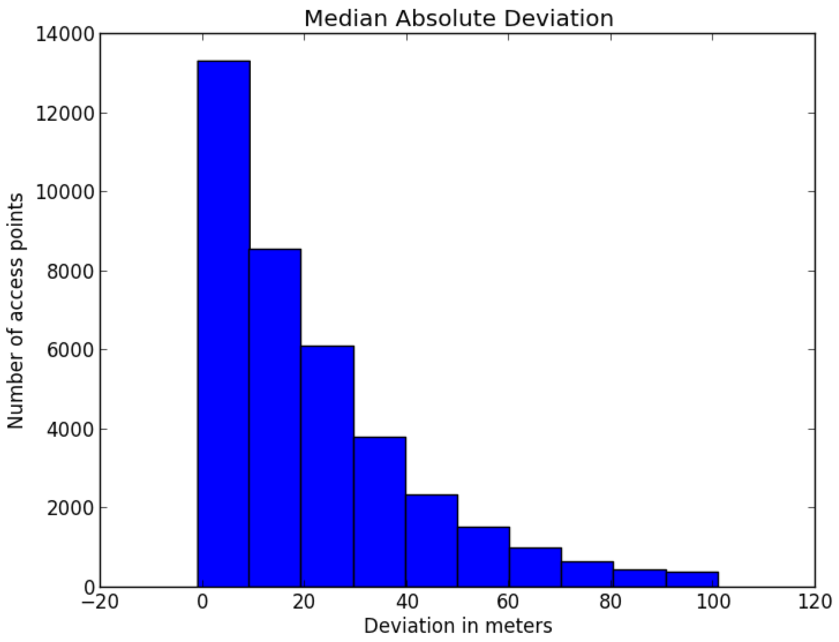


Figure 5.6: Histogram of access point deviations using the median absolute deviation

Other cities in Denmark, such as Odense, Aarhus and other smaller cities are also partially covered.

5.1.4 Revisiting the dataset of WiFi scans

On revisiting the full dataset of WiFi scans, it was found that **77 percent** of all of the WiFi scans performed in 4 months (this means about 14 million scans out of 18 million) contain **at least one** of the access points in the subset for which an accurate enough location estimation was determined. Therefore, even though the number of access points with an accurate location estimation is very low compared to the total unique access points detected, it has still covered a very large amount of the total of scans performed over 4 months. This means that using this set of access points it is possible to cover 77 percent of the places that students visit. It can also be inferred that students move 77 percent of the time within the locations determined by this small subset of access points.

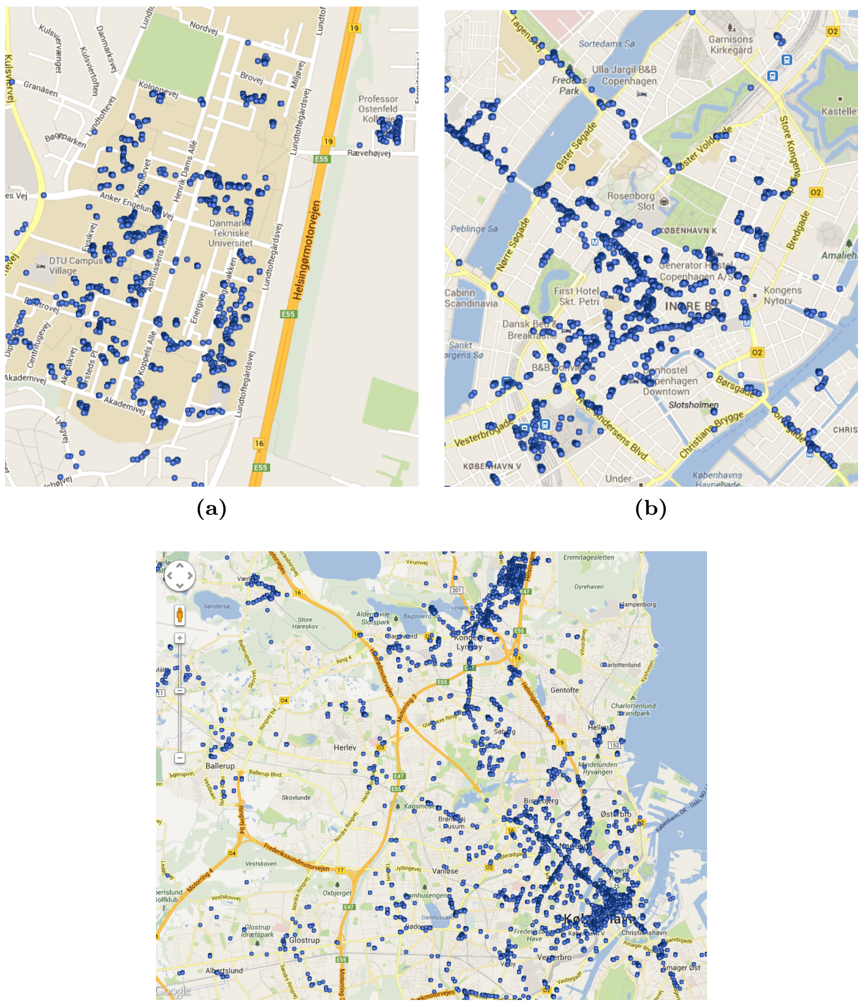


Figure 5.7: Geographical coverage of the access point subset with location estimates. DTU campus (a) and central Copenhagen(b) have a higher density of mapped APs. Greater Copenhagen region coverage is also illustrated (c)

This relates to[SQBB10] which has also studied human mobility using GSM towers and has concluded that 93 percent of the time, people move within the same areas. For the sake of comparing with this paper, if the threshold for the accuracy is set to 100 meters, **84 percent** of the total WiFi scans contain at least one of the trusted access points. This number is even closer to Barabaszi's, also keeping in mind that location provided by GSM cell towers can estimate location within more than 100 meters.

The estimation accuracy of the access points was evaluated also by comparing them to the locations provided by the Android location providers (GPS, WiFi and GSM cell tower providers), which is what the Funf location probe has used. The number of WiFi scans containing access points from the trusted set and that happened within 5 seconds of a location probe scan was of 12 000. The location estimation provided by the trusted access point was compared with the location provided by the Android location providers, by looking at the distance between them. It was found that **93 percent** of the location estimations provided by the trusted access points were maximum 30 meters away from the locations provided by the Android location providers. The histogram of the estimations which are at most 30 meters away from a Android estimation can be seen in Figure 5.8.

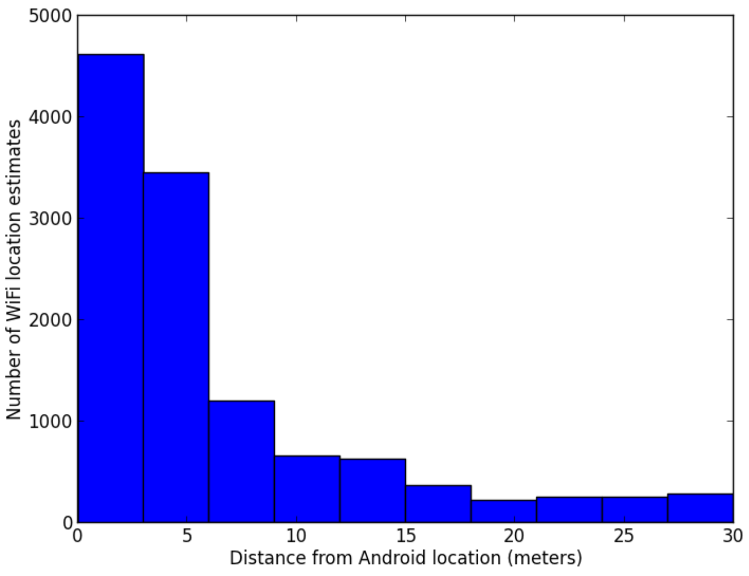


Figure 5.8: WiFi location estimates which are at most 30 meters away from an Android location estimation hapenning within 5 seconds of each other

By comparing the number of WiFi scans that could have provided a location estimate with an accuracy of 30 meters or better, with the number of locations provided using the Funf location probe (which uses the Android location provider), which means 13 million WiFi scans against 2 million location scans, we can obtain a temporal location resolution **6.5** times higher than the current one, without affecting battery life, within the areas where the trusted WiFi access points are present.

5.1.5 Validation using the DTU campus system

The access points in the final map that were corresponding to the DTU Campus WiFi system (identified by their SSIDs for the ones names "dtu", "eksamen" or "device" but also by their estimated location for the ones named "eduroam", since this network name is also used by other universities in Copenhagen) were compared to a list supplied by the IT department which mapped MAC addresses to building numbers. The distance from the location estimation of the access point to the building which it claimed to have been in, was computed. It should be noted the location for each building was considered the location of its center, and therefore errors of about 10 - 20 meters should be expected due to the actual positioning of the routers. The results show that about **60 percent** of the location estimations were less than 30 meters away from the real location of the building. Taking into consideration that access points could be 10 or 20 meters away from the center of the building, thus raising the accepted distance from the building to 50 meters, the results show that **85 percent** of the location estimations were satisfactory. About 3 percent of the locations were estimated with a deviation of more than 100 meters from the real location. The cause of this might be errors in the dataset, which, as stated by the campus IT department, is not completely up to date, and some of the access points might have been moved around campus. The total number of access points in the DTU campus system that were mapped is **2634**. In Figure 5.9, the histogram of the deviations from the real location of the DTU access points is illustrated. Figure 5.10 illustrates an example of a correct location estimation in the DTU campus system, while Figure 5.11 shows all of the access points in the campus system that have location estimates. As a side note, the coverage of the DTU campus presented here is different (contains less mapped access points) from the one in 5.7(c) because it does not include the access points belonging to users living in the on campus dormitory.

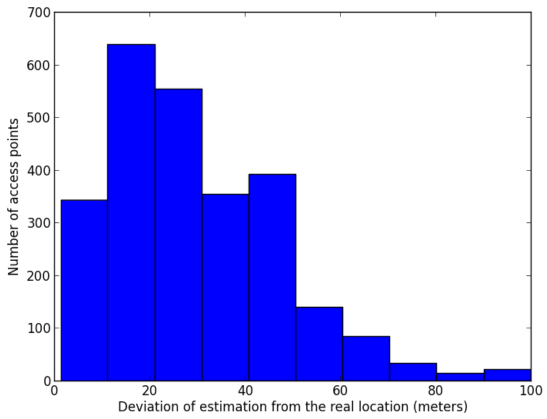


Figure 5.9: Histogram representing the deviation of the location estimates from the real locations for access points in the DTU campus system

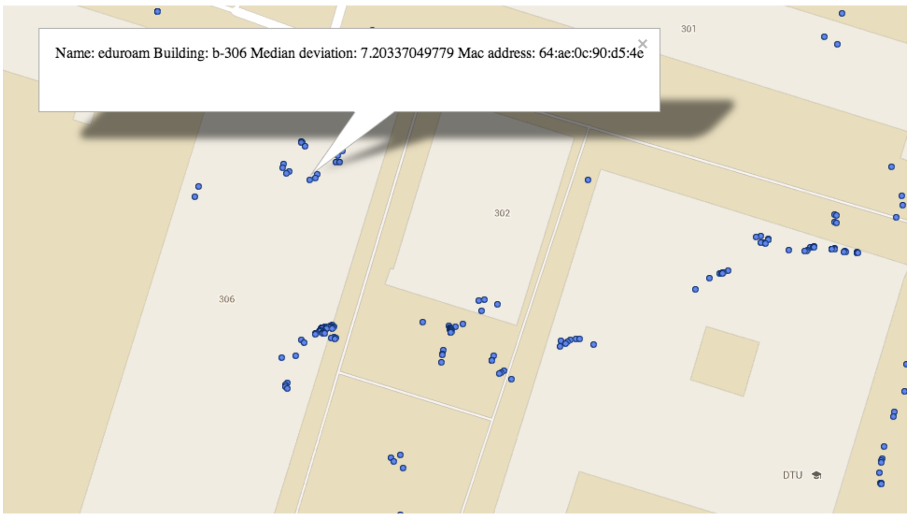

Name: eduroam Building: b-306 Median deviation: 7.20337049779 Mac address: 64:ae:0c:90:d5:4e

Figure 5.10: Location estimations for DTU WiFi access points

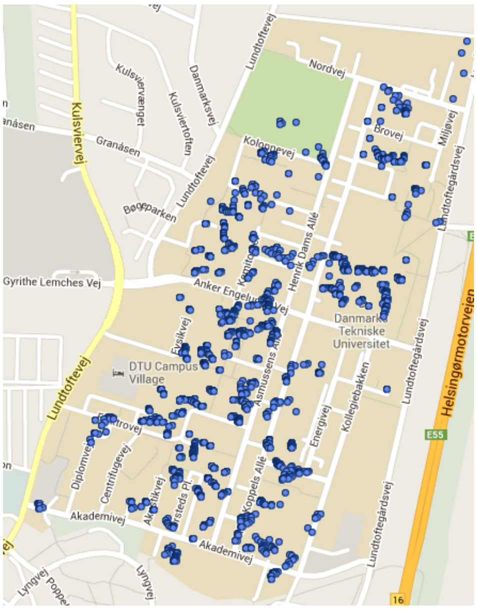


Figure 5.11: DTU campus system coverage

5.2 WiFi adaptive sampling in the SensibleDTU mobile application

The test setup used comprised of four configurations of the adaptive sampling scheme, run at the same time on different phones. The following configurations were tested:

1. Old configuration, with location scans being run every 15 minutes, abbreviated as "Old"
2. Adaptive configuration, using as a condition for location scanning the nearby presence of access points with a known location, abbreviated as "Trusted APs"
3. Adaptive configuration, using as a condition for location scanning the detection of user mobility, abbreviated as "Mobility"
4. Adaptive configuration, using both detection of user mobility and the nearby presence of known access points as a condition for location scanning, abbreviated as "Mobility+TrustedAPs"

Configuration	Location estimates per hour
"Old"	6
"Trusted APs"	58

Table 5.1: Impact of adaptive sampling on temporal resolution

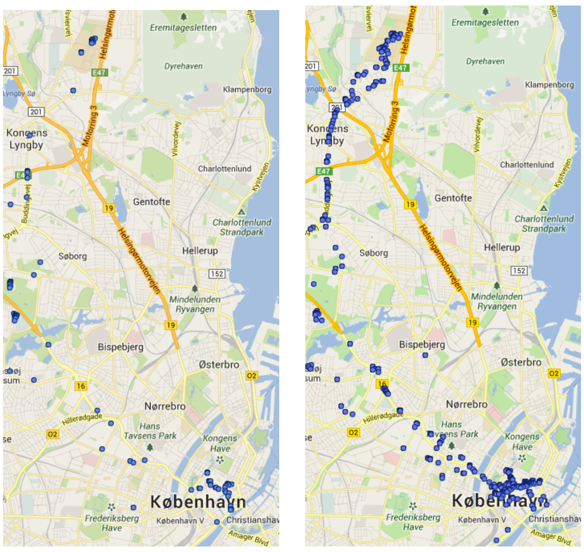
The application under test was the only application running on the phone while testing (except operating system processes). The areas covered for testing included the DTU campus, the city centre (since these places contain most of the mapped access points) and also places where there are no mapped access points. Travel was done by foot, by bus and by bike. The test cases also contained long periods of time when the user was static (so as to test the efficiency of the mobility recognition). Testing was performed for a total duration of 60 hours, and it tried to mimic the scenarios of a student that commutes to school or to the city by different means of transportation, and spends time during the day sitting and studying at university and visits the city.

5.2.1 Effects on location resolution

In order to establish the effects of the adaptive sampling scheme on the temporal resolution of location estimates, the number of location estimates provided per hour, calculated over the whole test period of 60 hours is presented in Table 5.1. The configurations compared are the "Old" configuration and the "Trusted APs" configuration since the latter one influences the location temporal resolution by providing estimates for detected access points. The "Trusted APs" configuration contains locations both estimated using WiFi scans and using the location probe.

The results show an increase of almost **10 times** in the number of locations provided over the whole tested period of 60 hours, which is an impressive increase in temporal resolution, considering that the sampling rates for the old configuration were not increased at all. It should also be noted that the "Old" configuration has an average of 6 locations per hour, instead of 4, as is expected considering the 15 minutes duty cycle of the location probe, due to location estimates received from other applications on the phone using the Android location provider. This behavior was not caused intentionally during the testing, but it was observed that the OS runs the "Maps" application in the background when the SensibleDTU application is running the location probe. It should also be noted that the "Trusted APs" configuration does not have implemented a means of retrieving locations from the passive location provider. Therefore, if just the locations retrieved by the SensibleDTU application would be reported, the number of locations estimated by the "Trusted APs" configuration would be about 15 times greater than the "Old" configuration.

In Figure 5.12, the location estimates obtained from the "Trusted APs" configuration is compared on a map with the "Old" configuration in order to better visualize the areas where a a high temporal resolution was obtained. The areas with a denser concentration of estimates are DTU and the city centre, due to the fact that these were the most frequented locations by students during the four months of study.



(a) (b)

Figure 5.12: Geographical distribution of location estimates when using configurations "Old" (a) and "Trusted APs"(b).

In Figure 5.13 a detailed view of the location estimates is presented, more exactly a day at DTU campus, to where the user has traveled by bike. An interesting aspect to notice here is that the "Trusted APs" configuration manages to capture the moments when the user is arriving and leaving DTU by bike (the area south of the campus), due to the higher temporal resolution of the WiFi scans. The user used one bike path for arriving at DTU and another one for leaving DTU. This aspect can only be distinguishable from the high location resolution one, thus proving the great advantage of having data of higher location resolution.

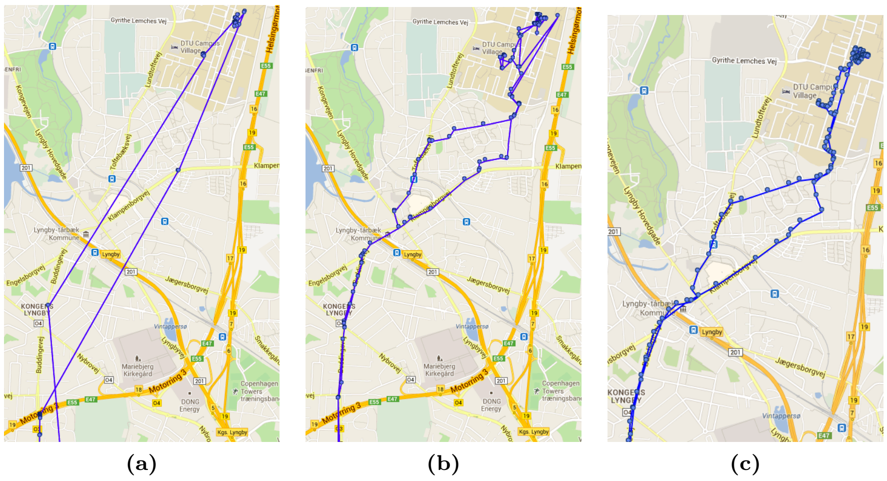


Figure 5.13: Detailed view of location estimates in the DTU campus when using configurations "Old" (a), "Trusted APs"(b) and ground truth (c). Lines represent user movement paths.

In order to validate the accuracy of the WiFi location estimates, the map was put side by side with the ground truth path (Figure 5.13(c)).

It can be observed in Figure 5.13(b) that there is a "jump" in the user's trail, to a place hundreds of meters further, which does not appear in the ground truth figure. This is due to the fact that the access point to which the user "jumps" very fast, might have been moved in time between mapping and testing, or it might have not been observed for a sufficient amount of time in the same position. This error seems to appear rarely though having shown up only 2 or 3 times in the approximately 3000 location estimates performed during testing. A closer look at the issue can be seen in Figure 5.14.

5.2.2 Effects on battery life

Battery performance of was evaluated using the average battery consumption per hour, measured on a scale of 0 to 100, 100 representing the maximum battery charge. Table 5.2 shows the average battery consumption of each of the four configurations over the whole period of testing (excluding, of course, values from the charging periods). The results show that the configuration which used only user mobility recognition as a condition for location triggering proved to be the most battery efficient overall. This can be explained by the fact that during the

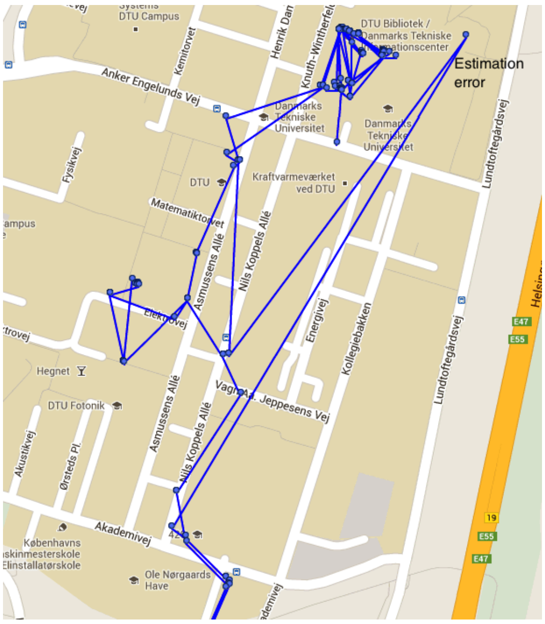


Figure 5.14: Error in location estimation caused by moved or incorrectly mapped access point

test period, the user has spent about two thirds of the time being static.

The most battery consuming configuration was not, as expected, the "Old" configuration, but instead the "Trusted APs" configuration. This (surprising) result might be explained by the different implementations of triggering the location probe. After testing two "Old" configuration with the two implementations for triggering the location probe, results have shown that the implementation described in 4.1, which is used by the adaptive sampling configurations, affects the battery 1.12 times per hour (3.53 battery units per hour vs 3.13 battery units per hour) more than the implementation used in the "Old" configuration. The cause appears to be the longer GPS run time (about 2 times longer) of the implementation used by the adaptive sampling configurations. This claim is partially supported by the number of times the location probe has been triggered by the "Trusted APs" configuration, higher than in the case of the other adaptive sampling configurations (as seen in Table 5.3). This explanation is, however, not consistent with the behavior of the "Mobility" and "Mobility+Trusted APs" configurations, where the "Mobility" configuration is more battery efficient than the "Mobility+Trusted APs" configuration, even though it has triggered the location probe more times (Table 5.3). The problem seems to lie somewhere in the

Configuration	**Battery consumption per hour (in battery units)**	**Improvement over "Old" configuration**
"Old"	2.85	-
"Mobility+Trusted APs"	2.53	12%
"Mobility"	2.41	18%
"Trusted APs"	3.12	-9%

Table 5.2: Battery consumption of different configurations

Configuration	**No. of calls to the location probe**
"Old"	190
"Mobility+Trusted APs"	96
"Mobility"	110
"Trusted APs"	161

Table 5.3: Number of times each configuration triggered the use of the location probe

implementation of the "Trusted APs" configuration, since it appears to affect the battery consumption of "Mobility+Trusted APs". Regardless of the problem however, the "Mobility" configuration can be used as an adaptive sampling solution, while still maintaining the improved temporal resolution of the "Trusted APs", since the collected WiFi is already collected and can be analyzed.

5.3 Performance of mobility detection

As seen in the previous section, the effects of mobility detection on battery life are quite significant. With regards to its performance in detecting user mobility, Figure 5.15 shows the scenario of a user sitting in an office for 2.5 hours. The "Old" configuration triggered the location probe a total of 10 times, while the "Mobility" configuration triggered the location probe only 3 times (which means that it detected user movement 3 times). This does not correspond to the ideal case (only 2 locations should have been provided, when entering and when leaving the room), however the outcome is significantly better for the battery, as the location probe gets triggered less. The error probably comes from higher fluctuations in the RSSI of the strongest access points as described in 2.

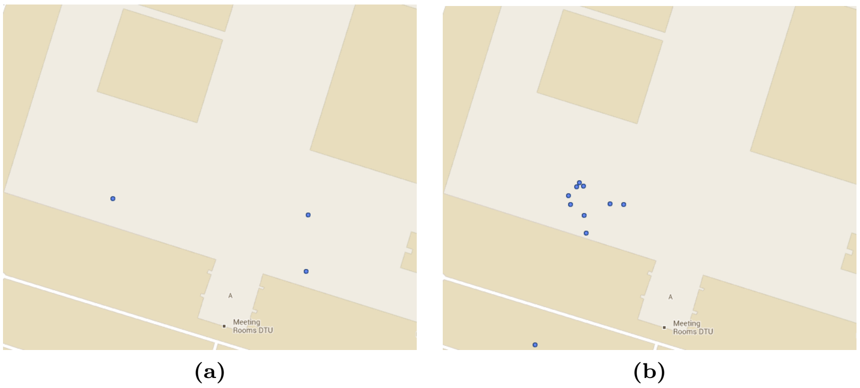


Figure 5.15: Location estimations when the user was static for 2.5 hrs. "Mobility" configuration(a) and "Old" configuration(b)

CHAPTER 6

Conclusions

The initial goal of this project was to increase the temporal resolution of the location data, while also improving the battery efficiency of the SensibleDTU application.

Regarding temporal resolution of location, the average number of **58** locations per hour, or one location almost every minute, seems quite satisfying. It should be noted however, that testing was performed primarily in areas with a high density of mapped access points, such as DTU or the Copenhagen central area. As was shown in section 5.1.4, more than half of the time is spent by the students in the places that have a high density of mapped access points, with DTU campus and central Copenhagen being the most dense areas. It is thus safe to conclude that temporal resolution of location estimates has been improved significantly.

The battery efficiency has also seen improvement. The best improvement observed by testing different adaptive sampling configurations is of 18%. This means an addition of about 2.5 hours more to the 14 hour battery life of the previous configuration (less than the proposed goal of 20 hours of battery life). This was obtained using a configuration that only prevents location scans using the GPS/network when the user is not moving, compared to an adaptive sampling configuration that does this by using both user mobility and the detection of access points known to the application, or a configuration that uses just the latter condition. The cause of this appears to be the under performance of the configuration that uses just detection of known access points, which affects the battery life more than the old configuration. The reason for why this happens is still open to discussion and further experimenting, but by judging by the number of times each configuration has accessed the GPS/network location providers, the combination of mobility detection and known access point detection should have been the most performant (with 14% less times requesting locations from the Android providers). Until this is resolved though, the mobility detection adaptive sampling scheme will be using for battery saving, while WiFi data can still be processed offline in order to obtain high temporal resolution.

A notable aspect encountered in the development of this project, was the creation of a map of access points in the Copenhagen area (and to a lesser extent Denmark), which can cover the locations of students 74% of the time with an accuracy of 30 meters or better, and 84% of the time with an accuracy of 100 meters or better. This is a quite an impressive result, considering the fact that the data was crowdsourced, with no special equipment, other than a smartphone needed, and no special behavior required from the experiment participants. It should be noted though that complete tracking of the users' mobility was only possible by also including GPS retrieved data, however, further improvement of the WiFi map should minimize the need for using GPS more, given the urban nature of the environment in which the experiment mostly takes place.

Chapter 7

Future work

The first and most important next step to be performed in this project is the re-implementation of the way adaptive sampling configurations request locations from the Android provider. It appears to be more energy consuming than the implementation of the previous configuration. This should improve the battery efficiency of the adaptive sampling scheme and it would also help shed some light in the unexplained behavior of the configuration using known access points detection. This is another area which should be investigated and experimented more, in order to establish why is it actually consuming more battery than the previous configuration.

Another feature discussed and designed in this paper was using the recently launched Google Location Provider which boasts a more efficient use of the battery. This should also be experimented on top of the other adaptive sampling schemes.

The creation process of the map of WiFi access points in the environment used in the experiment, should be improved by analyzing the number of days an access point was observed in (roughly) **the same location**. This would help to better identify access points that are mobile or have been moved. Also, the locations of access points should be re-evaluated constantly, in order to detect access points that have moved.

Another approach to reducing the usage of the location sensor is to use only one device to request locations, when more users are together in the same place. This method can make use of Bluetooth to detect when more users using the SensibleDTU application are in the same place and have just one of them (e.g. the one that has a higher battery charge) request for a location. This situation would be frequently encountered in the SensibleDTU experiment, since students are often in the same place (classrooms, library, bars, etc). In[Sap13] it is shown how WiFi can be employed to detect when users are together.

Further work should be put into implementing adaptive sampling framework-wide. Researchers should be able to easily implement conditional sampling for every probe in the Funf framework. Therefore, a way that can enable the developer/researcher to easily add and remove conditions for adapting sampling frequency would be quite useful. It could be something similar to the current way in Funf to configure sampling intervals and periods. Judging by the results obtained in this project, adaptive sampling in a generic way would prove to be quite an asset in a mobile sensing framework such as Funf.

Appendix A

Appendix

Bibliography

[API+11] Nadav Aharony, Wei Pan, Cory Ip, Inas Khayal, and Alex Pentland. Social fmri: Investigating and shaping social mechanisms in the real world. *Pervasive and Mobile Computing*, 7(6):643–659, 2011.

[CMT+08] Sunny Consolvo, David W McDonald, Tammy Toscos, Mike Y Chen, Jon Froehlich, Beverly Harrison, Predrag Klasnja, Anthony LaMarca, Louis LeGrand, Ryan Libby, et al. Activity sensing in the wild: a field trial of ubifit garden. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1797–1806. ACM, 2008.

[cod13] Rosetta code. Haversine formula, 2013. [Online; accessed 22-July-2013].

[DTU12] IMM DTU. Sensible dtu, October 2012.

[Fun13a] Funf. Developing new probes, March 2013.

[Fun13b] Funf. Location probe, March 2013.

[Goo13a] Google. Android location manager, July 2013.

[Goo13b] Google. Android services, July 2013.

[H+08] Ville Honkavirta et al. Location fingerprinting methods in wireless local area networks. *Master of Science Thesis, Tampere University of Technology, Finland*, 2008.

[Hen12] Robin Henniges. Current approches of wifi positioning. *SERVICE-CENTRIC NETWORKING - SEMINAR WS2011/2012, TU-BERLIN*, 2012.

[Hou11] Thomas Houston. Google's wi-fi database may know your router's physical location, April 2011.

[Inc13] Google Inc. Google now, July 2013.

[LML+10] Nicholas D Lane, Emiliano Miluzzo, Hong Lu, Daniel Peebles, Tanzeem Choudhury, and Andrew T Campbell. A survey of mobile phone sensing. *Communications Magazine, IEEE*, 48(9):140–150, 2010.

[MLF+08] Emiliano Miluzzo, Nicholas D Lane, Kristóf Fodor, Ronald Peterson, Hong Lu, Mirco Musolesi, Shane B Eisenman, Xiao Zheng, and Andrew T Campbell. Sensing meets mobile social networks: the design, implementation and evaluation of the cenceme application. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*, pages 337–350. ACM, 2008.

[Rac11] K Rachuri. K., mascolo, c., musolesi, m., rentfrow, pj (2011). sociable-sense: Exploring the trade-offs of adaptive sampling and computation offloading for social sensing. In *Proceedings of 17th ACM International Conference on Mobile Computing and Networking (Mobicom 2011). Las Vegas, USA*, 2011.

[RCK+10] Rajib Kumar Rana, Chun Tung Chou, Salil S Kanhere, Nirupama Bulusu, and Wen Hu. Ear-phone: an end-to-end participatory urban noise mapping system. In *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, pages 105–116. ACM, 2010.

[RMM10] Kiran K Rachuri, Mirco Musolesi, and Cecilia Mascolo. Energy-accuracy trade-offs in querying sensor data for continuous sensing mobile systems. In *Proc. of Mobile Context-Awareness Workshop*, volume 10, 2010.

[Sap13] Piotr Sapiezynski. Detecting face-to-face meetings using smartphone sensors. NetMob 2013 Conference, 2013.

[SQBB10] Chaoming Song, Zehui Qu, Nicholas Blumm, and Albert-László Barabási. Limits of predictability in human mobility. *Science*, 327(5968):1018–1021, 2010.

[SQL13] SQLite. Sqlite database, July 2013.

[TRL+09] Arvind Thiagarajan, Lenin Ravindranath, Katrina LaCurts, Samuel Madden, Hari Balakrishnan, Sivan Toledo, and Jakob Eriksson. Vtrack: accurate, energy-aware road traffic delay estimation using mobile phones. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, pages 85–98. ACM, 2009.

[Wik13a] Wikipedia. Geometric median, 2013. [Online; accessed 22-July-2013].

[Wik13b] Wikipedia. Mac address, 2013. [Online; accessed 22-July-2013].

[Wik13c] Wikipedia. Median absolute deviation, 2013. [Online; accessed 22-July-2013].

[Wik13d] Wikipedia. World geodetic system, 2013. [Online; accessed 22-July-2013].

[WLA+09] Yi Wang, Jialiu Lin, Murali Annavaram, Quinn A. Jacobson, Jason Hong, Bhaskar Krishnamachari, and Norman Sadeh. A framework of energy efficient mobile sensing for automatic user state recognition. In *Proceedings of the 7th international conference on Mobile systems, applications, and services*, MobiSys '09, pages 179–192, New York, NY, USA, 2009. ACM.