Vrije Universiteit Brussel

Faculty of Science
and Bio-Engineering Sciences
Department of Computer Science

An Extensible Framework for Indoor Positioning on Mobile Devices

Wouter Van Rossem

Promotor: Prof. Dr. Beat Signer

2011-2012

ABSTRACT

Recently, the domain of navigational applications has seen a huge rise. Due to the rising popularity of smartphones which are nowadays equipped with Global Positioning System (GPS) receivers, a great number of people who own these devices have the possibility to easily get information about their position and have a greater need for good navigational applications. Such applications, for example Google maps, work great when used outdoors. However, when we try to use these applications indoors, we have major problems in finding our correct position. Because the GPS needs to have an unobstructed line of sight to the GPS satellites, we encounter problems when trying to locate our position indoors.

Some theoretical aspects have to be covered, concerning radiolocation and indoor positioning, before we can continue by summarising the various different solutions that already exists in other projects, each with their own use of specific technologies and techniques. We will see that most of these were created for specific hardware and using some explicit algorithm however. And despite the fact that there exist a number of frameworks, they all have their own limitations.

Therefore, in this thesis we explore the notion of an Indoor Positioning System (IPS) in terms of a framework that works for various mobile devices and is extensible to be used with numerous algorithms and techniques. The design and implementation of this framework is discussed in detail and afterwards, an example IPS that uses Wireless local area network (WLAN) infrastructure and works with the Android platform is implemented using the framework. Several test were performed on the performance of this system, which are also presented.

ACKNOWLEDGMENTS

I wish to thank, first and foremost, my promoter Prof. Dr. Beat Signer for his continued support throughout the writing of this master thesis.

Secondly, I would also like to thank the company Go-Mobile for providing the opportunity to do an internship with them, which provided the basis of this thesis.

And finally, I would also like to thank André Miede for providing the beautiful LaTeXtemplate *Classic Thesis*[1].

1 http://www.miede.de/index.php?page=classicthesis

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

IPS	Indoor Positioning System
GPS	Global Positioning System
AOA	Angle of Arrival
TOA	Time of Arrival
TDOA	Time Difference of Arrival
RSS	Received Signal Strength
RSSI	Received Signal Strength Indicator
NN	nearest neighbour algorithm
k-NN	k nearest neighbour algorithm
API	Application programming interface
JDO	Java Data Objects
JPA	Java Persistence API
BSSID	Basic service set identification
SSID	Service set identification
RFID	Radio-frequency identification
NFC	Near field communication

WLAN Wireless local area network

DTDOA Differential Time Difference of Arrival

MLP Multilayer perceptron

XML Extensible Markup Language

UML Unified Modeling Language

JSON JavaScript Object Notation

1 INTRODUCTION

MOTIVATION Lately, there has been a greater need for location–aware and navigational applications. Almost all modern mobile devices such as smartphones and tablets come equipped with a GPS receiver, and more users own such devices. According to a survey done by Nielsen in November of 2011, 62% of the users in the age of 25 to 34 in the US, own smartphones[1]. Another survey, that was published by Google in January of 2012 shows that in the UK 45% of the total population owns smartphones. In France, this was found to be 38% and in Germany 23%[2]. So, while users can position themselves easily with these devices when outdoor, and use applications such as Google Maps for navigation or other location–aware applications, we still run into problems when trying to use these applications while indoor.

Since the GPS needs to have an unobstructed line of sight to the GPS satellites, it will not be possible to do positioning with this system when indoors. Therefore, other techniques have been proposed that use radio signals that can be used in an indoor location to do positioning there. The general idea is that by using properties of radio signals that are transmitted by devices at a fixed location (base stations), we can estimate a position of a device that can capture these signals. If we, for example, know the location of three base stations inside a building — and since we know that the radio signals propagate in a circle around the base station with diminishing signal strength — we can estimate a position for a device that is in the overlapping part of the circles by using some mathematical formulas. This is because we can look at the strength of the signals we receive and can estimate from this how far we are from each base station.

Numerous techniques such as this one have been explored in various projects, each having their own strengths and limitations, and most reaching good accuracy of around 1-2 m. The most limiting aspect about most systems however is that many were created to test a specific technique and were developed for a particular platform. And although a number of projects have provided systems that are more expendable for other techniques and may work on different platforms, they were still created for older and out–dated devices or are not expandable enough.

1 http://blog.nielsen.com/nielsenwire/online_mobile/generation-app-62-of-mobile-users-25-34-own-smartphones/

2 http://googlemobileads.blogspot.be/2012/01/new-research-global-surge-in-smartphone.html

For these reasons we have chosen to implement a generic and expandable framework for indoor positioning. Different building blocks are provided in the framework so that various techniques can be implemented, each with their own types of data that needs to be stored and algorithms that work with it. A clean separation between the data and how the data is stored is also provided, including a structured format for the data.

As an example of how the framework can be used, we discuss the implementation of an IPS that uses WLAN and is made to work with the Android platform. Two algorithms are also implemented and are tested by using a tool that is created specifically for this purpose.

OVERVIEW We begin in Chapter 2 by providing some theoretical background and by giving an overview of the different possibilities for implementing an indoor positioning system. It is divided into a discussion about the different technologies that can be used, followed by the different positioning algorithms that are available. Examples of existing systems are also provided to get an idea of how they work and how good they perform. Some details, and their limitations, of other existing indoor positioning frameworks are also handled.

Chapter 3 then provides a top–down overview of how the architecture of our IPS framework looks like and how it was implemented. We only handle the generic components that can be found in the framework, without going into detail yet of how it would be used for a implementing a specific IPS.

In Chapter 4 we discuss how we the actual framework can be used by describing the implementation of an IPS that uses WLAN. The different extensions that needed to be made and the two different algorithms that were implemented are described in detail in this chapter. We also discuss the applications that were developed for the Android platform and the tool that is used to test different algorithms and data setups.

In Chapter 5, we handle the different tests that were done using the tool to test the IPS at an indoor location we chose. The impact of different algorithms and data configurations will be the main topic of discussion.

Chapter 6 then finishes by giving some conclusions about the thesis and several possibilities on how the framework can be improved in future stages.

2 COMPARISON OF INDOOR POSITIONING SYSTEMS

Many different techniques and technologies have already been explored in other projects that implement an indoor positioning system. Therefore, in this chapter we provide an overview of the various technologies and algorithms that can be used to create such an IPS. First, we will give an overview of the theoretical aspects of an IPS such as what radiolocation is and how specific properties of signals can be used to do localisation. Next, an overview of existing algorithm types will be discussed followed by some examples of concrete systems than can be constructed using the mentioned signal properties and algorithms. Finally, we will also detail some of the existing indoor positioning frameworks.

2.1 RADIOLOCATION

When discussing localisation, both indoor and outdoor, we actually mean the more general process of radiolocation. Caffery and Stüber [4] define this as the measuring of parameters of radio signals that travel between a mobile station and a set of base stations, which are subsequently used to derive a location estimate.

They also identify two ways to implement a radiolocation system. In the first approach, the mobile station uses the signals transmitted by a base station to calculate its own position. This is the method that is used in GPS. In the other approach, the base station measures the signals transmitted by the mobile station, which are then processed by it or by a central site. The systems that we will discuss will mainly use the first approach for implementing a radiolocation system.

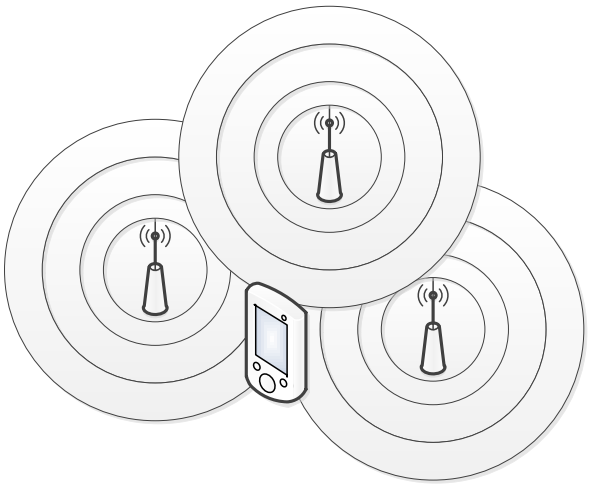
2.1.1 *Radiolocation Methods*

In the definition of radiolocation, we described that some parameters of the radio signal can be used to calculate the position. In the following we list the most interesting properties and their use:

SIGNAL STRENGTH The signal strength or Received Signal Strength (RSS) indicates the strength of the signal that is received from or by the base station. It is typically expressed in voltage per length or received signal power. The RSS can be used as a radiolocation method by using the known mathematical model that relates the path loss

attenuation[1] with distance. Therefore, the signal strength of the radio signal can be measured to provide a distance estimate between the mobile station and the base station. This is because we know that the mobile station must lie in a circle centred at the base station. If we use multiple base stations, the position of the mobile station can be calculated with greater precision [4].

Figure 1: Mobile station centred around three base stations



Two sources of errors that frequently arise with signal strength–based systems are defined by Caffery and Stüber [4]: *multipath fading* and *shadowing*. Multipath fading is a phenomenon that occurs due to constructive and destructive interference which causes the strength of the received signal to increase and decrease [29]. Shadowing presents itself when obstacles affect the propagation of the radio signal[2] which is important in indoor environments because, compared to outdoor locations, there can be many obstacles. This randomness of obstacles in the environment can be captured by modelling the density of obstacles and their absorption behaviour as random numbers [29].

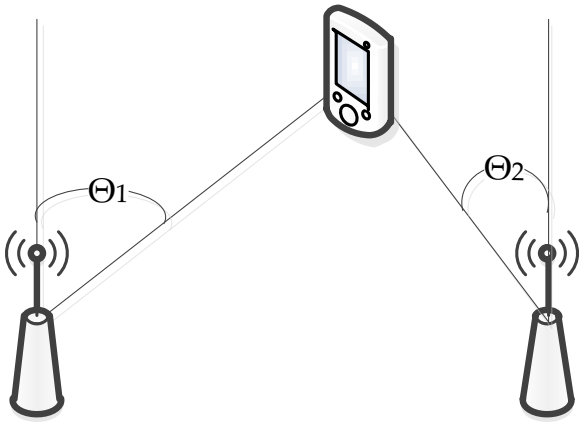
ANGLE OF ARRIVAL Another property of the signal that can be used is the Angle of Arrival (AOA) between a base station and a mobile station. This can be seen as an angle line formed by the circular radius from a base station to the mobile station. By using several of these angle lines, we can calculate the position by using the intersection of these angle lines [15]. Figure 2.1.1 shows an example of two such angle lines with angles Θ_1 and Θ_2.

TIME–BASED SYSTEMS The final property that we discuss are time–based systems that use the Time of Arrival (TOA) between a base

1 Attenuation is the amount by which an electrical signal weakens over distance as it moves though the transmission medium [16].

2 Radio propagation is the behaviour of radio waves when they are transmitted.

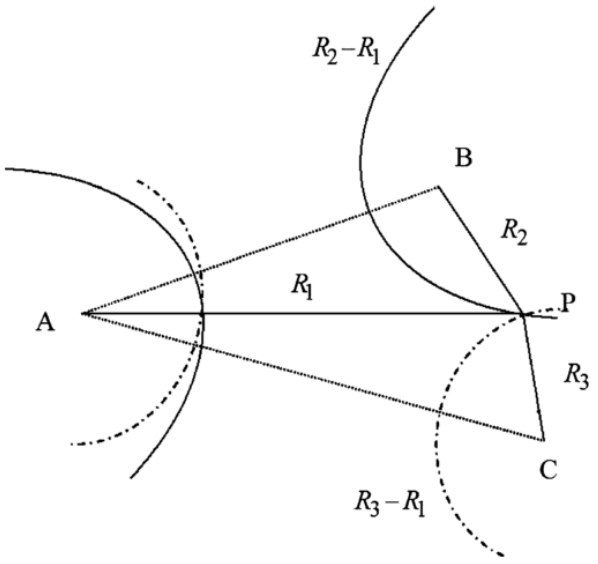
Figure 2: Angle of Arrival



station and a mobile station. Caffery and Stüber [4] distinguish two types of time–based systems: systems that estimate the TOA of a signal transmitted by the mobile system and received by multiple base stations and systems that use the Time Difference of Arrival (TDOA) of a signal received at multiple pairs of base stations.

In the first approach, we use the one–way propagation time between a mobile station and a base station. The distance between these two is directly proportional to the propagation time. In order to enable 2-D positioning, we would need the TOA measurement of at least three different reference points [15].

Figure 3: Time Difference of Arrival [15]



The second approach uses differences in the TOAs. Caffery and Stüber [4] explain the method: *"since the hyperbola is a curve of constant*

time difference of arrival for two base stations, the time differences define hyperbolae, with foci at the base stations, on which the MS [mobile station] *must lie"*. So the location of the mobile station can be calculated by using the intersection of those hyperbolae.

There are two problems that arise with time–based systems: all transmitters and receivers in the system must be precisely synchronised and the transmitted signal must be timestamped in order for the receiver to discern the time difference between sending and receiving of the signal [15].

2.1.2 *Positioning Algorithms*

Now that we have discussed some of the properties of signals that can be used and briefly touched on the algorithms that can be employed with these, we will take a closer look at how these measurements can be utilised to calculate the position estimate.

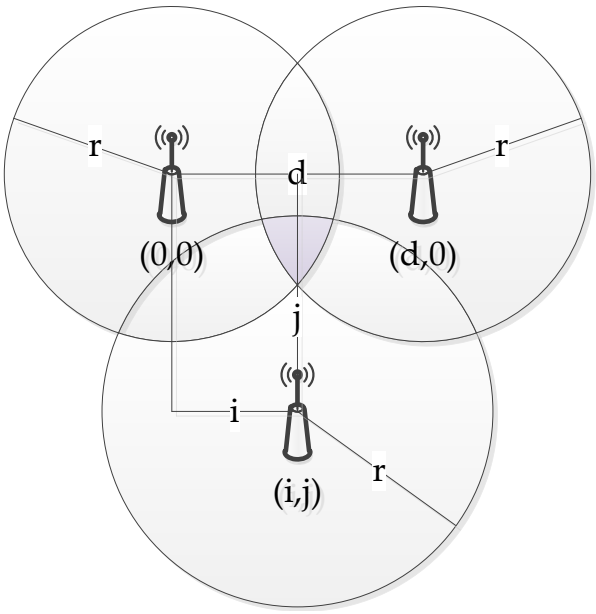
TRIANGULATION Triangulation is a method that uses the geometric properties of triangles in order to estimate the location. Liu et al. [15] make a distinction between two types of triangulation: lateration and angulation. In lateration, the estimate of a mobile station's position are calculated by measuring its distances from multiple reference points. The received signal strength can be used to calculate the distance, but often TOA or TDOA are measured. Figure 2.1.2 shows an example of trilateration with three base stations P1, P2 and P3, their respective coordinates and radii, which can be measured with a signal strength or time–based method. The position is then measured by a number of non–linear equations [5].

In angulation on the other hand, the location is estimated by calculating the angles relative to multiple reference points. A position estimate can then be found by using the intersection of several of these angle lines. As can be seen in Figure 2.1.1, the position of a target can be calculated by employing at least two known reference points and their two measured angles to the mobile station.

Some problems also exist with AOA techniques: the base stations need to be relatively complex in order to capture the angles — though no time synchronisation is needed between them — and the position estimate becomes less accurate when the mobile station moves far away from them. Furthermore, the same problems that are encountered with Signal Strength techniques — such as shadowing and multipath fading — also occur here [15].

SCENE ANALYSIS Scene analysis is a technique that can be divided in two separate stages: the *offline* and the *online stage*. In the offline stage, a site survey is done for the environment where we first need to collect so called *fingerprints*. These are measurements of the respec-

Figure 4: Trilateration example

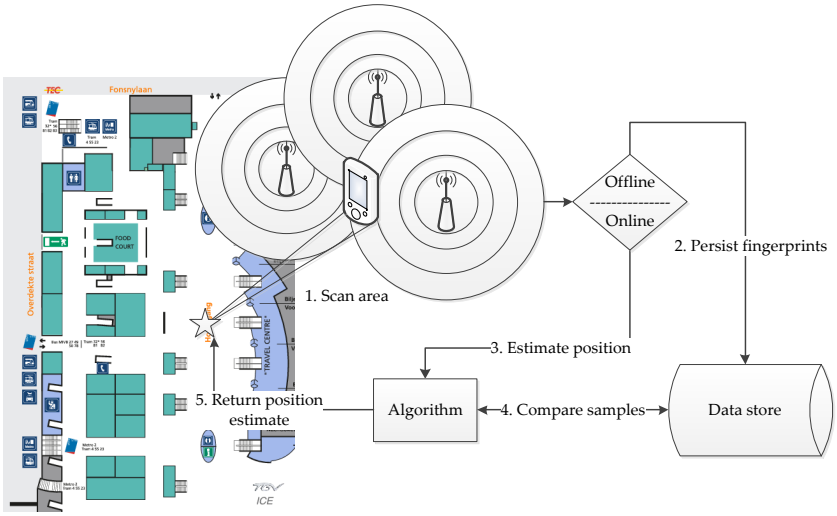


tive signal measurements from nearby base stations at a certain location. Most of the time this is done with an RSS based technique and fingerprints are collected for each location of a scene and stored in a database. During the online phase, we can then estimate the location of a mobile station by matching currently observed signal measurements and compare these, with a particular algorithm, to those that were collected in the offline stage. Some of the algorithms that can be used with this technique are the nearest neighbour algorithm (NN), probabilistic methods and neural networks [15]. Figure 2.1.2 shows an example of scene analysis. In step one, we scan the environment of a location on the map for the signal strengths of nearby base stations. If we are in the offline phase, we will then persist these measurements to the data store. In the online stage, the position can be estimated by using some algorithm, that will use the samples from the data store to do the calculations.

In NN we use the online measurement to search for the closest matches (fingerprints) from the database that was constructed in the offline phase. This is done by using some distance measure such as Euclidean or Manhattan distance. When using this technique it is also possible to create the database according to the root mean square errors principle [15].

Probabilistic methods consider positioning as a classification problem. If there are n location candidates $L_1, L_2, ..., L_n$ and m is a vector containing the observed signals from nearby base stations in the online phase, we can then use the following decision rule: "Choose L_i as estimated position if $P(L_i|m) > P(L_j|m)$ for $i = 1, 2, 3, ...n$ and $j \neq j$",

Figure 5: Scene analysis



where $P(L_i|m)$ means the probability that the estimated position is at L_i given the measurement m. Since $P(L_i)$ means the probability of being at position L_i and because we can assume that the probability of being on any position is the same, or $P(L_i) = P(L_j)$ for each $i, j = 1, 2, 3, ...$ Using Bayes' rule, we can therefore infer the following rule: "Choose L_i as estimated position if $P(m|L_i) > P(m|L_j)$ for $i = 1, 2, 3, ...n$ and $j \neq j$". Or, the probability that the signal vector m is measured, given that the mobile station is at position L_i [15].

For neural networks, a model such as a Multilayer perceptron (MLP) with one hidden layer is usually used. In the offline phase, the RSS values and their corresponding position are used as the inputs and the target during the training, after which the weights are obtained. At the online stage, an input vector with the signal strengths is multiplied with the trained input weight matrix. The result of this is put into the transfer function of the hidden layer neuron and whose output is multiplied by the trained hidden layer weight matrix. The output of the system is the position estimate [15].

PROXIMITY Proximity is a simple method that uses the location of the base station that is the closest to the mobile station as its location estimate. It requires a dense grid of base stations with well–known positions. When one base station detects a mobile station, it is assumed to have the same position. If there is more than one base station that detects the mobile station, it will be considered to have the same location as the one from which the strongest signal is received [15].

Now that we have discussed some of the properties of signals that can be used with certain algorithms, we have a look at how these can be used to implement an indoor positioning system. Each system that we discuss is based on one or more different technologies that are available in a smartphone and use some of the techniques that were discussed in Section 2.1.

Cellular–based Solutions

CELLULAR TECHNOLOGY The mobile cellular network can be used to estimate the location of the phone. Some towers periodically broadcast their location, which can be listened to and from which a location can be deduced [5].

For outdoor positioning, the accuracy of this method is generally low (in the range of 50-200 m), but can be higher in densely covered areas. Indoor positioning is possible if the building is covered by several base stations or one base station with a strong RSS that can be received by indoor mobile clients [15].

SYSTEMS AND SOLUTIONS A GSM–based indoor localisation system was introduced by Otsason et al. in [20]. They make use of so–called *wide* signal strength fingerprints, which include the 6 strongest cells that are traditionally used in the GSM standard and also makes use of cells which are strong enough to be used for efficient communication. The position estimate is calculated with k nearest neighbour algorithm (k-NN) and, during experiments, was shown to have a median accuracy of 5 m and furthermore supports multi-floor buildings.

Wireless Local Area Network

WLAN can be used in an IPS since each access point sends out periodic broadcasts. From this signal we can get some information such as RSS and Signal to Noise and also information about the access point itself such as the Basic service set identification (BSSID), which acts like an individual identifier for the beacon. Passive scanning is used to listen for signals from the base stations [5].

WLAN FINGERPRINTING The WLAN fingerprinting consists of the two stages that were discussed in the scene analysis: the offline and online stages.

In the offline stage, we calibrate the area where positioning needs to be conducted. This process involves manually traversing a building with a Wi–Fi enabled device which is constantly taking snapshots of the signals from detectable access points at each location [5]. A

database is then constructed where each location corresponds to a different RSS snapshot.

In the other stage, these RSS snapshots can be used to compare the detected RSS values at a certain location. The corresponding coordinates of the pattern that closest matches the observed signal can then be assumed to be the location of the device.

SYSTEMS AND SOLUTIONS RADAR from Microsoft was the first project to use Wi–Fi signal strength for an IPS. They also introduced the idea of fingerprinting and combined this with a radio propagation model. A nearest neighbours algorithm was used, with the Euclidian distance as a metric. The median error distance they observed in their system was between 2 and 3 m [1, 23].

Another project is *Horus* which claims to have a high accuracy (less than 1 m on average) while maintaining low computational requirements by using clustering techniques. Positioning is then done by a probabilistic method [30].

Battiti et al. [2] on the other hand propose a positioning method using neural networks (more specifically, a MLP). By using a learning algorithm and a set of labelled samples a model can be built that can, when confronted with new data, generalise. An average accuracy of 2-3 m was observed in their system.

An important distinction can also be made between the previous projects: indoor positioning systems that try to model the radio propagation and those that do not. For the former the positions of the base stations are needed, while for the latter this is not required.

Radio–Frequency Identification

RFID TECHNOLOGY Radio-frequency identification (RFID) is a technology that uses radio waves to transmit data from an RFID tag through an RFID reader. RFID tags can be attached to an object and consist of an antenna, a transceiver and a small amount of memory. The RFID reader has more functionality than a tag and in addition to an antenna and a transceiver it also contains a power supply, a processor and usually an Ethernet or serial interface to connect to a network. By using the reader, we can track or identify a tagged object [5].

There are two types of RFID tags: active and passive tags. Passive RFID tags operate without a battery. They reflect the radio signal transmitted to them from a reader and add some information by modulating the reflected signal. The range for this however is very limited, ranging from one to two meters. Active tags on the other hand have a small power supply. This allows them to actively transmit in reply to an interrogation by a reader and ensures that they have a much better coverage, in the range of tens of meters [15].

SYSTEMS AND SOLUTIONS LANDMARC [18] is an example of a system that uses active RFID tags for locating objects inside buildings. A number of RFID readers are installed at specific locations so that the whole area can be divided into sub–areas, where each area can be identified by a subset of the RFID readers that cover that sub–area. Given an RFID tag, an area can then be found for that tag based on the subset of readers than can detect it. In order to increase the accuracy without having to add additional expensive readers, they also utilise fixed tags that help to calibrate the positioning. This approach does require the signal strength from tags to readers. The position estimate is then calculated using the k-NN and has an average of around 1 m.

A combination between RFID and WLAN is also proposed by Spinella et al. [27]. The performance of the WLAN fingerprinting approach is improved by combining RFID technologies. By using this hybrid approach, they tried to solve the phenomenon where, in the offline phase, very similar Received Signal Strength Indicator (RSSI) values are captured in different areas. The RFID tags are placed in the different zones of the area to differentiate between this ambiguity. In the online phase, the RSSIs are measured of both the WLAN access points and RFID tags. A general zone for the mobile station is first calculated by taking the zone of the RFID tag with the highest RSSI value. Afterwards this zone is used when calculating the position estimate with the WLAN fingerprints. Average location errors of less than 1 m were achieved by using this approach.

Bluetooth

Bluetooth technology is a short–range communication technology with the key features of robustness, low power consumption and low cost. The Bluetooth specification defines a uniform structure for a wide range of devices to connect and communicate with each other. The range depends on the class of radio that is used and can range from one meter (class 3 radios) to one hundred meters (class 1 radios) [3].

For implementing an IPS using Bluetooth, we can use Bluetooth tags, which are small transceivers[3] with a unique ID that can be used to locate the Bluetooth tag [15].

CELL IDENTITY Every Bluetooth device has a unique identifier which can be used for indoor positioning. When a Bluetooth device connects to a piconet, it receives the IDs from the other devices in the network. This ID can then be used to look up the position of the device in a database. Alternatively, the device can send its own position after the connection is established.

3 Devices that can both transmit and receive communications.

When there are multiple base stations in range, these can be used to calculate a more precise location since the user can only be located in the overlapping part of the range of the base stations.

Tags are laid out inside a building and great care must be taken in positioning the base stations. There needs to be sufficient overlap of the ranges, while the individual overlapping areas should be small in size [11].

RECEIVED POWER LEVEL The problem with cell identification however is that we only get an accuracy of several meters at best [11]. Received power level on the other hand gives additional information on the position of the user since it is proportional to the distance from the sender. Also see Section 2.1.1 for more information about signal strength.

In the Bluetooth system, we receive the power level as RSSI, which is only a relative measure. It can however be converted to an absolute power level if the Golden Receive Power Range[4] of the particular receiver is known. This is not a fixed value too, but we can use the Transmit Power Level which is sent by a Bluetooth device to other users by default.

SYSTEMS AND SOLUTIONS A system is introduced in [6] by Feldmann et al. that uses RSSI from Bluetooth beacons to derive an approximation of the distance. The actual localisation is then done by a trilateration method. An average error of around 2 m was observed in their tests.

Fischer et al. [7] on the other hand used a TDOA technique in their Bluetooth indoor positioning system. The time synchronisation issue is avoided by utilising a technique they call Differential Time Difference of Arrival (DTDOA). In this technique, there is one master base station that transmits a signal to the mobile station and the base stations. When this signal is received, an internal clock is started on these devices. These clocks are then stopped by a signal that is sent by the mobile station. The position can be estimated by using the measured time differences and taking into account the propagation time between the master and the other base stations.

Near Field Communication

NFC TECHNOLOGY Near field communication (NFC) is a short–range radio technology, in the range of four centimetres. Communication is started when two NFC–compatible devices, such as smartphones or NFC tags are within close proximity [19]. Because of this very low range, the only real possibility is by using some proximity technique.

4 An optimal power level for limiting battery consumption during transmission [8].

	RFID	BLUETOOTH	NFC
SET UP TIME	$< 0.1\text{ms}$	$\pm 6\text{s}$	$< 0.1\text{ms}$
RANGE	$\leqslant 3\text{m}$	$\leqslant 30\text{m}$	$\leqslant 10\text{cm}$

Table 1: Summary of short range technologies [19].

Table 1 presents an overview of the previously discussed RFID, Bluetooth and NFC short–range technologies in terms of set up time and range [19].

SYSTEMS AND SOLUTIONS Ozdenizci et al. [21], for example, presented a system that uses NFC to implement an indoor navigation system. An NFC tag is placed at the entrance of a building that contains the URL which can be used to download the indoor map. In the mobile application, a user can then select a destination for which the application will calculate the shortest route. The user is navigated to their destination, with tags scattered around the building that can be used to verify the current position of the user or redirect them.

Systems Summary

Table 2 and Table 3 show a summary of the different indoor positioning systems that are possible. For each type of system, we list the types of methods and their corresponding possible algorithms. The other properties such as how accurate the system is on average, how complex it is in general to set up and what the expected cost can be is based on observations of some of the previously mentioned existing systems.

The easiest methods to set up are the ones using scene analysis since, unlike with triangulation methods, we do not need to model the propagation and need no information about the location of mobile stations. On the other hand, scene analysis needs a major manual effort in collecting the fingerprints.

In terms of costs, we can see that systems using WLAN are the least expensive ones since commonly the environment is already set up with several WLAN access points. On the other hand, for Bluetooth, RFID and NFC the respective tags still need to be purchased and placed.

Precision varies between most systems, with good results having been achieved with different approaches, so no real best system can be identified.

	CELLULAR	WLAN	NFC
Methods	RSS	RSS	Cell–id
Algorithms	Triangulation, Scene Analysis	Scene Analysis	Proximity
Accuracy	Low	Medium to High	High
Complexity	Medium	Medium	Medium
Cost	Low	Low	Medium

Table 2: Summary of Indoor Positioning Systems (1)

	BLUETOOTH	RFID
Methods	RSS, AOA, Time–based	RSS, AOA, Time–based
Algorithms	Proximity, Triangulation	Proximity, Triangulation
Accuracy	Medium to High	Medium to High
Complexity	Low to Medium	Low to Medium
Cost	High	High

Table 3: Summary of Indoor Positioning Systems (2)

2.3 EXISTING FRAMEWORKS AND CONCLUSIONS

Although there exist a lot of solutions for IPS as can be seen from the previously mentioned projects, there is still a lack of generic frameworks for implementing an IPS on modern mobile devices. There are some examples though which we discuss here.

Marcu et al. proposed a positioning framework for WLAN in [17] with low power usage as their main objective. They used a trilateration technique but should also be able to use scene analysis. Not much information is given about the actual use of the framework and how extensible it is.

Another framework, named *ipos*, is presented in [22]. It provides "*a configurable, scalable, and open architecture for easy integration into third party applications*". The framework has a number of components of which the actions will be performed in the following order: capturing of measurements independent of the wireless technology (offline phase of fingerprinting), preprocessing on the data found in the first step, location estimation, postprocessing of the estimated location such as trajectory prediction. The framework was developed by IMST GmbH but does not seem to be available any more.

Tran et al. [28] also implemented an indoor positioning framework. It is divided into several separate components: a *preprocessor* for the signals before they are captured, a *calibration engine* for capturing and labelling the data, a database with signal signatures for each location,

an optional module for describing geographical information of an indoor location and a module to estimate the position. An optional module is also introduced that keeps track of neighbour locations of other locations for faster retrieval.

Two indoor positioning frameworks for Android were found: Smart Space [14] and Airplace [12]. Not much information about the Smart Space framework is provided except for a presentation from 2009 given at the Droidcon convention[5]. It should support WLAN fingerprinting combined with some motion detection, but it is not clear how expandable it is for other techniques and algorithms. Airplace on the other hand is a new platform from 2012 for WLAN fingerprinting on Android devices. It provides an application for collecting fingerprints and finding the position of a user. In the online phase, the application works by downloading the complete radio map so that users can locate themselves independently afterwards. This however means that all the positioning estimating logic is executed on the Android device. At the time of writing, the platform was not available for download via the project website[6].

In general, we can conclude that although there exist some indoor positioning frameworks, most have some limitations: some are outdated, are no longer available and were not made for modern mobile devices. Others are too focused on a specific technique or do not provide information of how they can be expanded. Therefore, we decided to create our own generic and expandable indoor positioning framework that is geared towards modern smartphones.

5 http://de.droidcon.com/dc2011/de/article/smart-space-indoor-positioning-framework

6 http://www2.ucy.ac.cy/~laoudias/pages/platform.html

3 IMPLEMENTING AN IPS FRAMEWORK FOR WLAN

In this chapter, we first discuss some of the choices that were made in creating our framework for indoor positioning. After that we continue by giving a complete overview of the design and implementation of the framework.

3.1 INTRODUCTION

As discussed in the previous chapter, we found that, although there exist numerous different projects that implement some sort of indoor positioning system and even found some frameworks for indoor positioning, there is still a need for a generic and extensible framework that is also tailored to modern mobile devices. Therefore we want to have a framework that works for different kinds of devices and with all sorts of positioning techniques. We begin our discussion of the framework by formulating some goals that our framework needs to achieve. After that we start by giving a top–level view of the framework and will keep expanding on this with more details.

Goals

The limitations that we found in other projects and frameworks were the basis for the goals that we propose in order for our our framework to separate itself from other projects. The goals are the following:

GENERIC Our framework needs to support as much techniques as possible without having to do a lot of drastic changes. Basic building blocks need to be available that can be reused and expanded for other systems.

DATA STANDARDISATION Data capturing is an expensive task and the captured data should therefore be in a standardised format so that it can be reused and is not directly coupled to the framework and the used data store.

INTERCHANGEABLE ALGORITHMS In order to have flexible systems, algorithms should be as interchangeable as possible.

Assumptions

Before we continue with detailing the actual framework, we would like to note that indoor positioning is a broad research field with

all kinds of methods being used that may be incompatible with our framework. Therefore, we would like to point out that the main focus of the framework is WLAN scene analysis or fingerprinting approaches since these have proven to be the most popular and most effective methods for doing indoor positioning on mobile devices. Whenever possible however, other methods were taken into account and the framework was made as generic as possible at those points. Due to the nature of the framework, only methods where the mobile station takes some measurements and sends it to a central site for processing are considered though. Because when we would do the computations on the mobile device, this would make it platform dependent again.

The framework is also written in Java due to its cross–platform nature and widespread use. The design of the framework are therefore done in an object–oriented manner and using Unified Modeling Language (UML) diagrams.

3.2 FRAMEWORK ARCHITECTURE & DESIGN

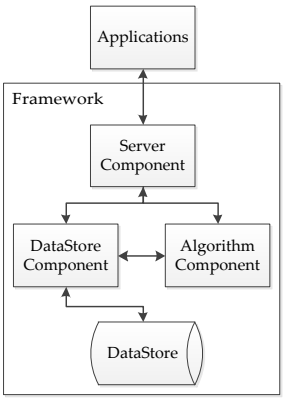
We now start by giving a top–level overview of the framework without going into implementation details yet, and continue by explaining the design of the framework using UML diagrams.

3.2.1 *Components Overview*

In order to fulfil the goals we proposed for our framework, we separated it in different components. Figure 6 shows a top–level overview of the overall architecture. At the top, we can see the applications that make use of the framework. These can be applications on mobile devices or even other applications that for example use the framework to do batch testing of algorithms. Below this, we can see that all communication with the framework is done through the *server component*, which will process requests for position estimates or other other functionalities of the framework. The server component itself uses two other components: the *data store component* and the *algorithm component*. The former is required for handing all operations with the data store when data that needs to be retrieved or saved and is also used by the latter to run different algorithms in various configurations. Next, we discuss these different components in more detail.

DATASTORE When creating an indoor positioning system, there is always the need to store some data. For scene analysis methods these can be the fingerprints that need to be stored or, for triangulation methods, this can be the positions of base station and so forth. In our framework we therefore have a dedicated data store component that accomplishes the following goals:

Figure 6: Top–level overview of the framework



- Provide basic data entities that are used in most indoor positioning systems, and can be expanded for specific scenarios.
- Save the data in a standard format so that it can be easily imported, exported and reused.
- Decouple the component from the actual data store that is being used in order to allow any type of underlying data store.

This allows for a very flexible data store component which is necessary since data collection is an expensive task.

ALGORITHMS The other important part of an indoor positioning system are the algorithms that are run to calculate the position estimate. This is usually done by using some measurements of the radio signals as captured by the users on their mobile device and some data from the data store. For this we have an algorithm component which achieves the following:

- Provide a generic interface for algorithms to implement and therefore make them as interchangeable as possible.
- Allow for specialisation for particular techniques.

This allows us to have algorithms that are easily interchangeable.

SERVER In order for the users to be able to position themselves, we need to have a central site to which the users can send their measurements and let it be processed so that a position estimate can be calculated. Additionally, it also supports the uploading of new data to the data store by providing an interface for this. The server component carries this out:

- Allow the user to make positioning requests and return the calculated position.

- Allow positioning requests to be configurable in terms of what algorithms to run, how parameters of algorithms should be set and what should be returned.
- Support uploading of new data to the data store so that it can easily be automated.

As we can see, the server component handles all communication between the client applications and the framework.

3.2.2 *Data Flow*

Before we continue to give the design details concerning the framework, we take a closer look at the data flow diagram in Figure 7 that shows what happens when making a positioning request from a mobile device. The mobile device sends a positioning request, containing the captured measurements and the algorithm configurations, to the server component. The server then first configures the algorithm component based on the parameters in the request and then lets the algorithm calculate the position estimate. The algorithm component might also use some data from the data store for calculating the estimate. Once a position is calculated, it is returned to the client that made the request.

Figure 7: Flow of a positioning request

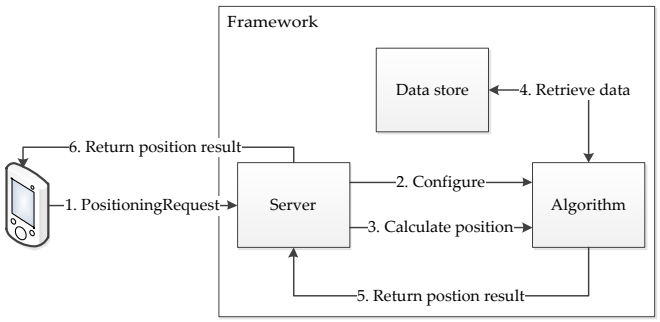
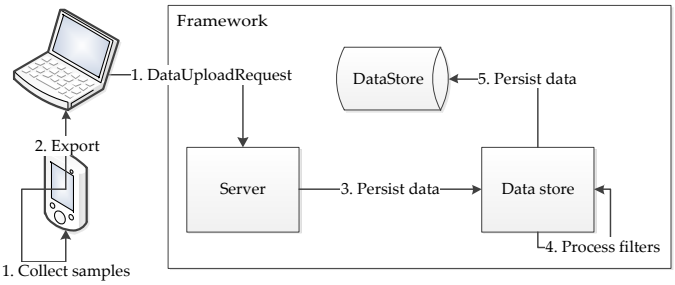


Figure 8 provides a top–level overview of the capturing of data and the following upload request with the data. Data is first captured on the mobile device and then exported to, for example, the user's laptop from where a request is sent to the server to upload the new data. The request can also contain some filters that will be processed by the data store component before persisting the data. These can for example filter out some noisy values.

3.2.3 *Design*

Now that we have seen how the architecture looks like from a high level, we go more in depth by providing details on the design of

Figure 8: Flow of a data upload request



the framework. We first begin by describing the data building blocks that are provided by the data store component and are used in most indoor positioning systems. After that we will see how the algorithms were made interchangeable and what the architecture of the server component looks like.

Data Entities & Manager

In order for our framework to be generic, we have a number of basic data entities that are present and allow for implementing different types of indoor positioning systems, and can be expanded upon for other types of systems. The data entities that are available and provide the basic building blocks are:

BASE STATION A device which can be uniquely identified and sends out some signal that can be captured and from which certain properties can be observed.

SIGNAL MEASUREMENT A measurement of certain signal properties with the captured measurement values. For each property there can be a number of captured values since this allows us to for example take the average of the values or get other relevant information (`SignalMeasurementValues`).

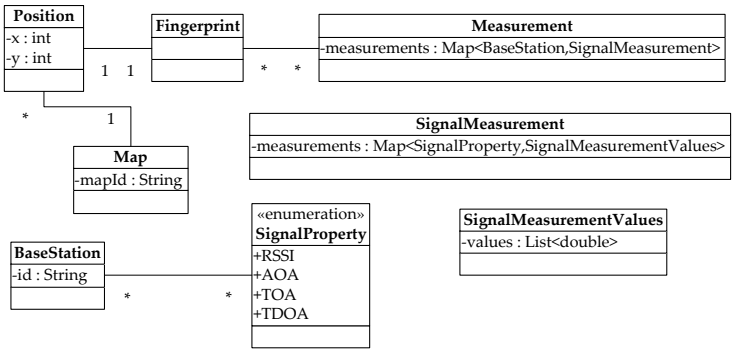
MEASUREMENT Contains the different `Signal measurements` of certain `Base stations`.

FINGERPRINT A `Measurement` at a certain `Position`.

POSITION A simple 2-D position on a map where a `Measurement` can take place or that is returned as a position estimate by a positioning algorithm. It is defined by its X and Y coordinate on a map of an indoor location.

This can be modelled in a class diagram as can be seen in Figure 9. Another thing to note is that a measurement has a map containing base stations and their corresponding signal measurements, and that

Figure 9: UML Diagram of the database entities



a signal measurement itself has a map holding a list of values for each signal property.

All communication with the actual data store where these data entities are stored, is done by using a `Data Manager`. The actions that it supports are:

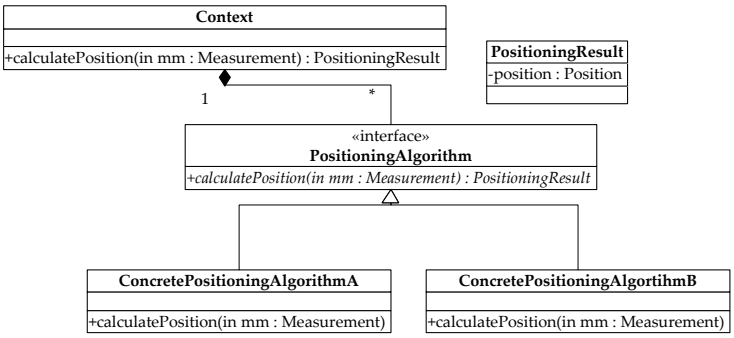
- Query for measurements of certain base stations.
- Query for all fingerprints, fingerprints at a specific location or having a measurement with a particular base station.
- Persist new measurements and fingerprints.
- Update existing measurements and fingerprints.

Algorithms

One of the important aspects about our framework, is that we want to have our algorithms as generic as possible so that we can interchange them. Therefore we have opted to use the strategy pattern which allows us to define a family of algorithms, encapsulate each one, and make them interchangeable. The algorithms can also vary independently from the clients that use them [9]. Figure 10 shows the UML diagram of the strategy pattern for our positioning algorithms with the main components: `PosioningAlgorithm`, `ConcreteAlgorithm` and the `Context`. The `PositioningAlgortihm` declares an interface that is common to all the algorithms and for which the `ConcreteAlgorithms` provide implementations for separate algorithms. The `Context` is then configured with a `ConcreteAlgorithm` and uses the interface to call the algorithm.

The method `calculatePosition` defined in the interface `PositioningAlgorithm` takes a `Measurement` as captured by a user and returns a `PositioningResult`. In its most basic form, this `PositioningResult` contains the estimated `Position` but can be expanded to contain additional information, depending on the algorithm.

Figure 10: UML Diagram of the strategy pattern for positioning algorithms

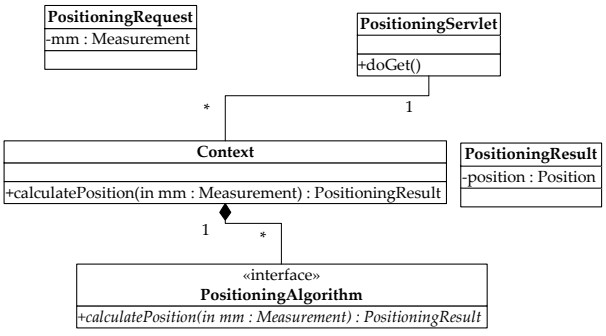


Server

For our server we need to have two servlets: one for positioning requests and another for data upload requests. The former uses the algorithm component as discussed in the previous section to configure which algorithm the `Context` should use and how it should be configured. The position estimate is then calculated by the concrete algorithm. While the latter is used to process requests to upload data to the data store. This is done by using the data manager which allows for data such as fingerprints and measurements to be persisted and can also filter some values by using so called data upload filters. These can be simple mathematical comparators to check if some value of the signal measurement is true for this filter, and if so needs to be filtered out.

Figure 11 & 12 shows the architecture of the server in UML diagrams.

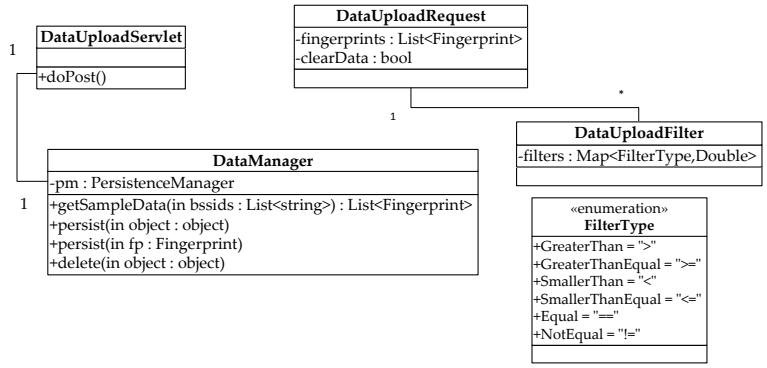
Figure 11: UML Diagram of the server architecture (1)



3.3 ARCHITECTURE IMPLEMENTATION

Now that we have discussed the design of the framework, we continue by providing details about some of the implementation choices that were made and have an impact on how the framework works.

Figure 12: UML Diagram of the server architecture (2)



One of the goals of our framework is that we want our data store to be flexible in terms of what underlying database is used while also have standardised data that can be easily exported and imported. In this section it will become clear how we achieved this.

DATABASE Our data store needs to save measurements and fingerprint samples from the offline phase of the scene analysis that are collected. Because we do not want to be tied down to one specific type of database, we have chosen to use Datanucleus. This is a project that provides management of data in a Java environment. It supports the Java Data Objects (JDO) (and also the Java Persistence API (JPA)) Application programming interface (API) that allows mapping to a wide range of data store types (RDBMS, ODBMS, Map–based, Web–based, documents, etc.). In our framework, we have chosen to use a relational data store, namely a MySQL database. However the flexible nature of the Datanucleus platform would allow us to use any of the dozens of data stores that are supported[1]. We have also chosen to use the JDO API since this allows for a wider range of data store types, while JPA is limited to only relational data stores.

Using Datanucleus is very simple once the dependencies are resolved. A properties file needs to be created where the data store driver and the connection parameters are specified. The Java classes that need to be persisted can then be defined by using annotations.

```
@PersistenceCapable
public class Measurement {

    @Persistent(table = "BS_MEASUREMENTS")
    @Join(column = "BASESTATION_ID")
    @Key(types = ips.data.entities.BaseStation.class)
    @Value(types = ips.data.entities.SignalMeasurement.class)
    protected Map<BaseStation, SignalMeasurement> measurements;
}
```

1 For a full overview of the supported data stores, see `http://www.datanucleus.org/products/accessplatform_3_0/datastores.html`

Apart from that it is also necessary to have a constructor that takes no arguments and getters and setters for each attribute of the class. After that, it is possible to persist objects to the datastore using the `PersistenceManager` and query for objects in the datastore. Several query languages, including SQL, are supported but the special JDOQL query language is preferred since it is object–oriented and database agnostic.

DATA REPRESENTATION Since the capturing of data will take place on a mobile device, we also want to have an intermediary format for storing the data before persisting it to the data store. For this we have chosen to use the Extensible Markup Language (XML) format because of its widespread use and the fact that it is both easily readable by humans and machines.

In our framework, the XML representation is generated by using the *Simple XML* library[2]. Similarly to Datanucleus, it requires the classes to be annotated as can be seen in the following example:

```
@Root
public class Measurement {

    @ElementMap(entry = "measurement", attribute = false)
    protected Map<BaseStation, SignalMeasurement> measurements;
}
```

Once this is done, we are able to *serialise* a Java object from the annotated class to an XML representation and *deserialise* an XML entity into a corresponding Java object.

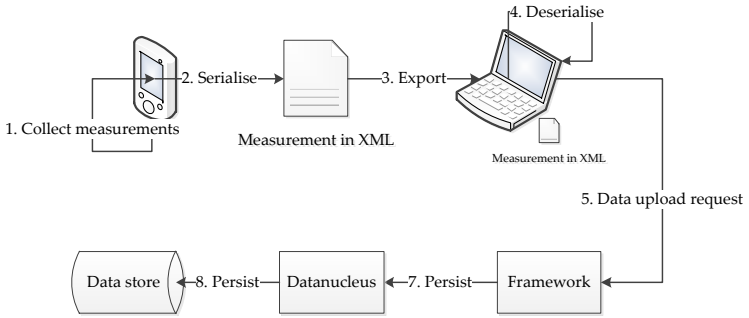
Another alternative format to store the data would have been to use the JavaScript Object Notation (JSON), but because of the capabilities of the Simple XML framework, this was not further pursued.

COMBINING By combining DataNucleus and Simple XML we have an interesting way of exporting and persisting data. Measurements from the mobile device can be serialised to XML — by using Simple XML if Java is supported or by creating the XML with some other library — and can afterwards be deserialised so that it can be persisted to the data store. Figure 13 provides a graphical overview of how this works. Measurements are first collected on the mobile device and are serialised to XML using the Simple XML framework, after which they can be exported. That XML document can then be deserialised again to get the original measurement as a Java object, after which a data upload request can be made with this measurement. The framework will then persist the object by using Datanucleus.

SERVER The last implementation details that we need to discuss are about the server component that needs to receive positioning and

2 http://simple.sourceforge.net

Figure 13: Datanucleus and Simple XML data upload process.



data upload requests. For implementing this component, we have chosen to use the Tomcat server. Two servlets are created to receive the requests using HTTP GET. When a request is made by the client, it needs to include the request (which includes the measurement and configuration parameters) as an XML object. The servlet reads the serialised XML `PositioningRequest` or `DataUploadRequest` and deserialises this again into a Java object using the Simple XML framework. From a positioning request we then get the measurement on which to run the algorithm to calculate a position estimate, the algorithm type to use and optional parameter settings for the algorithm. These are used to configure the algorithms through the `Context`, after which the position estimate can be calculated. From a data upload request we can get all the data that needs to be uploaded and the filters, all of which will be forwarded to the `DataManager` so that it can be persisted.

3.4 CONCLUSION

In this chapter we have given a complete overview of the system by starting with the design of the system and providing implementation specific details. And now that we have a better understanding of the framework, we can take a look at a more in depth summary of the different requests that can be handled by the server.

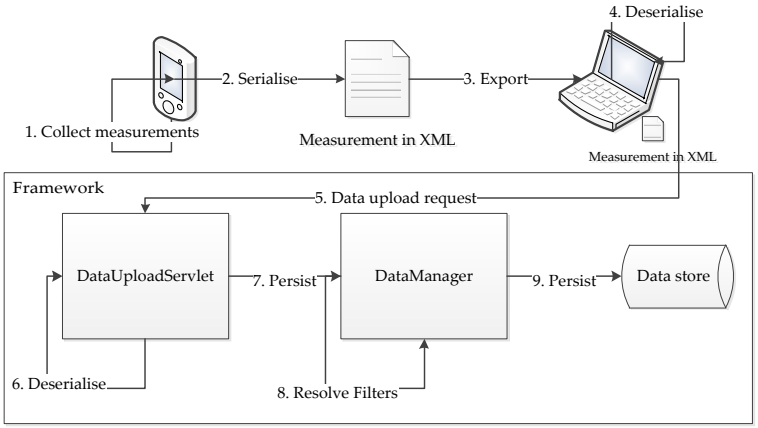
DATA UPLOAD FLOW Figure 14 shows the complete flow of a data upload request. The different steps are:

1-2 `Measurements` are collected on the mobile device and serialised to XML using the Simple XML framework.

3-4 The serialised measurement is exported to the user's laptop where it can be deserialised again.

5 The measurement can be sent as a `DataUploadRequest` with possible parameters. This request will be serialised and sent as an HTTP PUT request to the server.

Figure 14: Data upload request flow



6-7 The request is received by the server and the measurements and parameters are extracted and forwarded to the `DataManager` which will persist them.

8-9 The data manager will resolve any filters and use Datanucleus to persist the data to the data store.

POSITIONING REQUEST FLOW Figure 15 shows the process of calculating the indoor position in the context of a fingerprinting method. The steps are as follows:

0 The smartphone or other mobile device first does a scan of its surrounding environment in order to for example capture the detected signal levels of nearby access points, this is the `Measurement`.

1a A new `PositioningRequest` is created with the captured measurement. The request contains some parameters that are set in the application such as what algorithm to execute etc. This is serialised using Simple XML and sent as an HTTP GET request from the device to the server.

1b The server receives the request with the serialised XML. This is deserialised again using the Simple XML framework so that we retrieve the `PositioningRequest` and its `Measurement` from it.

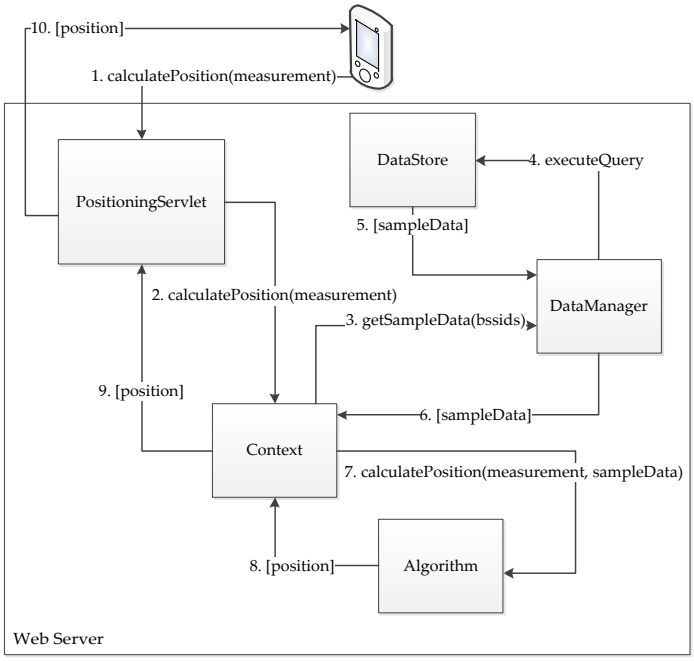
2 The algorithm that is specified in the `PositioningRequest` is configured through the `Context` and called to calculate the position estimate with the measurement.

3-9 The algorithm will retrieve some sample data from the database using the `DataManager`. This will run a query through Datanucleus to the data store to for example retrieve the fingerprints

that share an access point with the measurement. After that the positioning algorithm can calculate the position estimate.

10 This position estimate is then sent back as a *serialised* `PositioningResult` to the user who made the initial request, after which the application that made positioning request can do something with this position estimate.

Figure 15: Server data flow



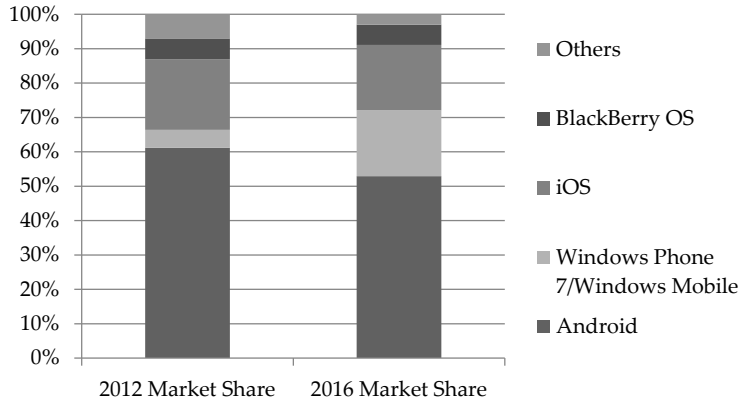
4 USING THE FRAMEWORK

In this chapter we provide an overview of how the framework can be used by developers to add new techniques or algorithms. This is done by providing an example of how we use the framework to implement an IPS using a WLAN fingerprinting method.

4.1 MOBILE PLATFORM

A first important choice that arises is the device that will be used for implementing the IPS since this will impact the kind of sensors that can be used and therefore the types of techniques. Also, because we want to have a prototype application on a smartphone, a number of platforms are available: iOS, Android, Windows Phone and BlackBerry OS. These are the most popular mobile operating systems according to the IDC Worldwide Mobile Phone Tracker statistics[1]. Figure 16 shows the smartphone operating system market shares of 2012 and the expected numbers for 2016.

Figure 16: Worldwide smartphone operating system 2012 and 2016 market share



Not all technologies in the mobile device are always readily usable on a platform. Table 4 gives an overview of the technologies that can be used by the previously listed mobile platforms. A check mark in the table means that information about the technology can be accessed by some API provided by the platform. This information can, for example, be that we can get the RSS from a base station. RFID was

1 International Data Corporation (IDC) does a quarterly worldwide survey of smartphone operating system usage. At the time of writing, the latest survey was from June 6th 2012 http://www.idc.com/getdoc.jsp?containerId=prUS23523812

not included in this table since in general there are almost no mobile devices equipped with an RFID reader. From this table we can see that Android and Blackberry provide the easiest access to information about these technologies, while iPhone and Windows Phone do not make this information accessible due to security reasons.

	CELLULAR	WLAN	BLUETOOTH	NFC
ANDROID	✓	✓	✓	✓
IOS	✗	✗	✗	✓
WINDOWS PHONE	✗	✗	✗	✓
BLACKBERRY OS	✓	✓	✓	✓

Table 4: Summary of usable technologies on different mobile platforms

We chose to use the Android platform for a number of reasons. The first and most important being that, compared to for example iOS and Windows Phone 7, Android allows easier access to low level APIs such as the WLAN which will be crucial for our implementation. On iOS it would also be possible but only after jailbreaking[2] the iPhone. Another reason being of a more practical nature: the fact that we already had an Android phone at our disposal[3] and we already had some experience developing on the platform. As can be seen in Figure 16, the Android platform currently also has the highest market share amongst operating systems on smartphones, which is another important factor.

RADIOLOCATION METHODS For the technology that we will use with the mobile device, WLAN was chosen. This is because it provides the easiest option by using infrastructure that is already in place. Bluetooth could also be used, but to use it we would need to have access to Bluetooth tags which would need to be purchased and placed in an indoor location. RFID was not an option since most (if not all) Android devices do not have an RFID reader.

The fact that Android allows such low level access to the wireless stack that provides Wi–Fi network access, also make it a great choice. Functionality of the `android.net.wifi` package includes reading all information from the current connected network as well as initiate a scan for Wi–Fi access points and terminating or initiating Wi–Fi connections. We can also easily get the RSS from the found access points, which is necessary for fingerprinting. The only other mobile operating system that was found to have such APIs is the BlackBerry OS

2 Jailbreaking is a process of using hardware or software exploits to remove some of the limitations imposed by Apple on iOS devices.

3 The Android phone that was used was a Samsung Galaxy S II running Android 4.0.3

from RIM[4]. However, due to the much greater popularity of Android, it is still the more attractive option.

ALGORITHMS Since we are using WLAN, the most straightforward algorithm to use with this technology is scene analysis or fingerprinting. We also decided to implement two different algorithms to calculate the position estimate: nearest neighbours and a statistical method. These two algorithms can then be compared against each other, while using the same samples and thus minimising the manual effort that is needed in the offline stage.

4.2 EXTENDING THE FRAMEWORK

In order to implement an IPS that uses WLAN fingerprinting, we extend the framework in two ways: the addition of entities specifically for WLAN fingerprinting and the two algorithms to calculate a position estimate.

4.2.1 *Data Entities*

In order to support WLAN fingerprinting, the basic data entities from the framework need to be extended so that we can save data specific to WLAN fingerprinting. These are the entities that need to be added:

ACCESS POINT A type of base station used in WLAN that allows other wireless devices to connect a wired network. It is identified by its BSSID (address), has a human readable name or Service set identification (SSID), a single string denoting the capabilities of the access point (for example what type of security it supports), the frequency in MHz of the channel over which the client is communicating with the access point and the detected signal level in dBm. This is generally the information that can be found when doing a WLAN scan on a mobile device.

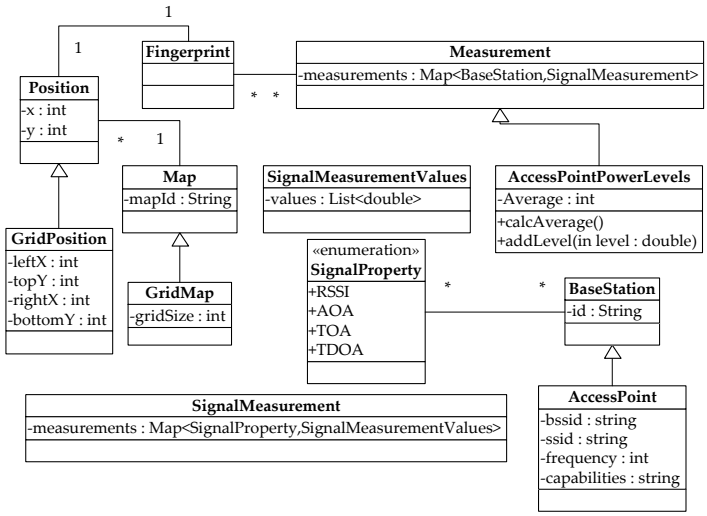
ACCESS POINT POWER LEVELS A type of `Signal measurement` with measurements from only the signal property `RSSI`, that is the different power levels of nearby access points that are captured from WLAN scans.

WLAN–FINGERPRINT A fingerprint used with WLAN that is defined by its position and the `Access point power levels` that are captured at that position.

4 `http://www.blackberry.com/knowledgecenterpublic/livelink.exe/fetch/2000/348583/800451/800563/What_Is__Network_Diagnostic_Tool.html?nodeid=1450596`

We can see that generalisation is used to extend the basic building blocks for a WLAN fingerprinting approach. Figure 17 shows the complete UML diagram with the new entities.

Figure 17: UML Diagram of the WLAN database entities



4.2.2 *Algorithms*

We decided to implement two algorithms that work with WLAN fingerprinting. For both cases, we assume that we have a sufficient amount of samples, i.e. WLAN fingerprints, that are collected and stored in the data store. The process of collecting these samples, using a dedicated application, will be discussed later.

A UML diagram of how these algorithms were introduced in the algorithm component as discussed in Section 3.2.3 can be seen in Figure 4.2.2. The two algorithms implement the `PositioningAlgorithm` interface and are therefore *concrete algorithms*.

Nearest Neighbour Algorithm

The first algorithm that we implemented is the nearest neighbour algorithm that uses a distance measure to find a position estimate.

As we discussed in Section 3.2.3, a positioning algorithm is started when the sever receives a position request which contains the measurement that the user captured and some configuration options. This measurements includes the RSSI values of the surrounding access points. From these, we can use the BSSIDs to get fingerprints from our data store, namely all the fingerprints that have an access point with a BSSID equal to an access point with the same BSSID from the measurement. This means that that the measurement and the fingerprint share some similarity and is thus relevant to be used to calculate the

distance between the two. Other possibilities would be to fetch samples that contain exactly the same access points or a fixed number of equal access points.

Just as in a regular nearest neighbour algorithm we would then like to calculate the distance between our instance (the measurement) that needs to be classified and our sample data. To calculate this distance we use the Euclidean distance:

$$d(\mathbf{m}, \mathbf{f}) = \sqrt{\sum_{i=1}^{n} (f_i - m_i)^2}$$

In this case m is our measurement and f is a fingerprint. Another distance measure that can be used is the *Manhattan distance*.

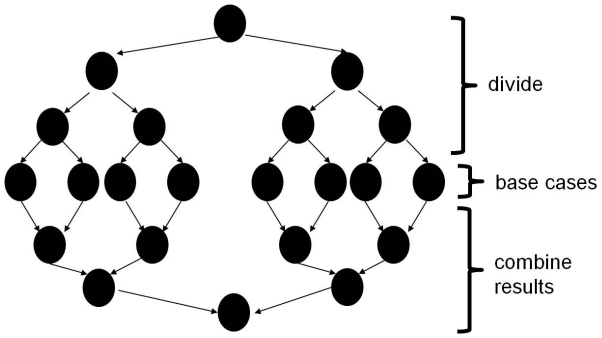
We then go sequentially over all the access points of the measurements and calculate the distance between the power level of the access point of the measurement and the average power level of that access point of the fingerprint.

One important aspect to note here is that the distance depends on the number of equal access points in the measurement and the fingerprint. In general, the more access points that are matched, the more precise our approximated position will be. This is not true in all cases however and if we would just return the position of the fingerprint with the most matched access points and the smallest distance, we would get some incorrect results. Therefore we first sort the list of distances on the number of matched access points and then on the distance. From these we pick a fixed number of elements from the front of the list and take the element with the smallest distance as our calculated position.

One thing to note as well is that our nearest neighbours algorithm uses the *ForkJoin* framework that was introduced in Java 7[5]. This allows us to have a parallelised version of our nearest neighbour algorithm. The way it works is that the work to calculate the distance between the sample and each fingerprint from the list of samples we retrieved is always divided until we reach a cut–off point where we will thus calculate the distance between the measurement and a subset of the fingerprints. The map of `Map<AccessPoint,DistanceResult>` that is calculated by each base case is then combined until we get the complete result. The ForkJoin framework itself is responsible for managing the threads that calculate the distances. The sorting of the distance results is then done with *QuickSort*, which is also implemented by using the ForkJoin framework.

5 It can also be used in Java 6 by using the JSR-166 library.

Figure 18: The ForkJoin process [10]



Bayesian Method

The previous algorithm for localisation that we described is a type of *deterministic approach*. Yim et al. [13] describe another way of calculating the position, i.e. by using a *probabilistic method*. In this method we will use the Bayes rule:

$$p(p_t|m) = \frac{p(m|p_t)p(p_t)}{p(m)} \tag{1}$$

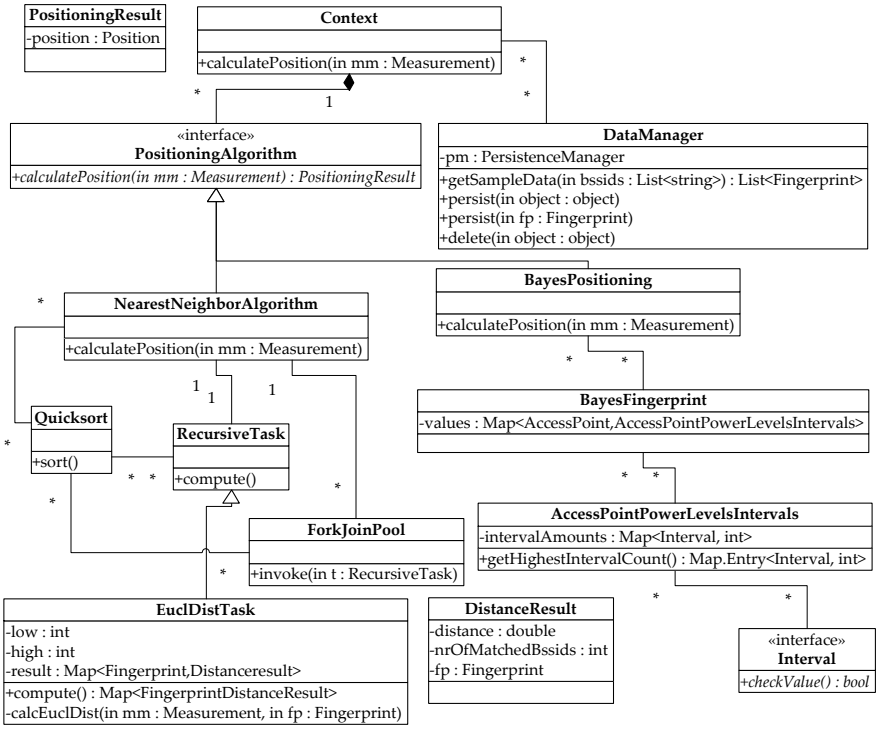
Or in other words: the probability of being at position p given measurement m is equal to the probability of observing m at position p, and being at position p in the first place. We now calculate this probability of being at location l given the observation from our mobile device for each fingerprint in our database. The algorithm will then predict that the position is p if $p(p_i|m) > p(p_j|m)$ for $1 < j \leqslant m$, $j \neq i$ where m is the number of possible samples. And since $p(m)$ is constant for all measurements, we only need to maximize $p(m|p_t)p(p_t)$.

In the most basic case we can also say that $p(p_t)$ is constant, since it can be on any position. If we would take into account the previous locations of the user we would need a more sophisticated method to calculate this probability. One such method is given by Markov Localization [26]. Here we say that $p(p_t)$ is *"the 'transitional probability' from all locations at* $t-1$ *to* 1 *at the current time* t*, multiplied by the probability of being at those locations at* $t-1$*"* [13]. Or:

$$p(l_t) = \sum_{l_{t-1}} p(l_t|l_{t-1})p(l_{t-1})$$

Where we know $p(l_{t-1})$ from the previous localisation attempts and $p(l_t|l_{t-1})$ is calculated using a *motion model*. In our case, this would be how far we expect the user to move between t and $t-1$. By using these methods, we could remove some of the noise affected outliers from the output.

Figure 19: UML Diagram of the algorithms



4.3 CLIENT APPLICATIONS

In order to test our IPS, two applications were created: an Android application to collect fingerprint samples and to send positioning requests, and a Java tool that allows for batch testing of the algorithms.

4.3.1 *Android Applications*

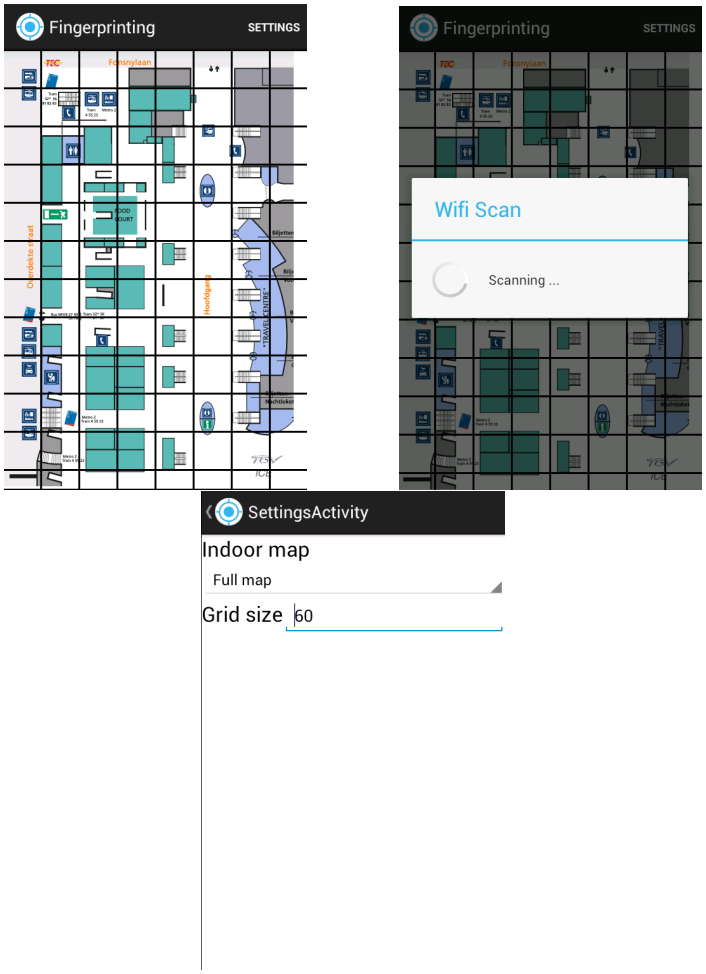
The Android application is divided in two parts: one is used for gathering sample data in the offline phase and the other is used for the online phase when doing the actual indoor positioning.

GRID CELLS For our Android application, we have also chosen to do an extension of the `Position` entity and added the entity `GridPosition`. Instead of using the absolute position for a fingerprint, we divide our map of the indoor location in different configurable grids of the same size. A grid position is then defined by its top left and bottom right X and Y coordinates. This will allow us to do some interesting tests by modifying the grid sizes and viewing the impact on the IPS. Figure 22 shows an example of grid cells on a map of an indoor location.

OFFLINE APPLICATION The Android application for the offline phase has an image of the map (using an `Imageview`). On that map

there is a grid of cells overlaid that represent the different fingerprint locations. The size of these grids can be configured and will be useful to test the impact of grid size on the accuracy of the positioning algorithm. When collecting samples, we can also get multiple samples for a same grid and combine them afterwards.

Figure 20: Android application for offline phase of fingerprinting

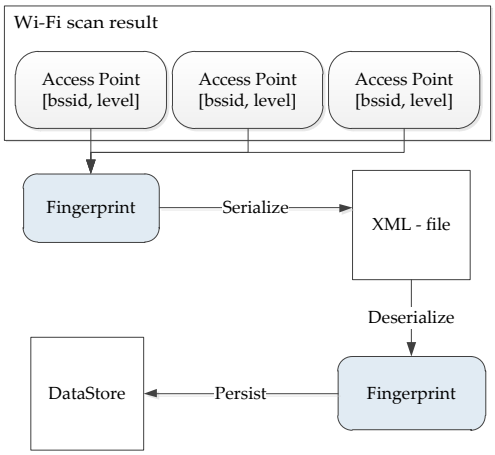


The activity[6] implements the `OnTouchListener` interface which allows it to receive touch events. When the user touches the map, the Wi–Fi scan is initialised by using the functionality provided in the Wi–Fi package. For that grid location where the user pressed, we store these results from the Wi–Fi scan in an XML document. The *Simple XML* library is used for this, which will serialise the WLAN fingerprint we captured. Figure 21 gives a graphical overview of this sampling process.

There is also another activity that acts as simple settings menu where the user can select different maps to do fingerprinting on, and modify the grid size.

6 An Android application component that provides a screen with which users can interact in order to do something

Figure 21: The sampling process



ONLINE APPLICATION For the online stage a separate application is provided which looks similar to the offline one, in the sense that it also has an ImageView that shows the map of the location. In this application however, we can initiate a positioning requests by using a button in the menu. When this is pressed, we will also start a Wi–Fi scan. Now the results of this scan are however sent to our server instead of being saved to the SD card, where the localisation algorithm will be run and which will return our estimated position. The results are sent to the server in the XML format using the Apache HTTP Client. When we have received the location, a pin will be drawn on the map denoting this location estimate. The is also a simple settings menu to choose which positioning algorithm to run for the request.

4.3.2 *Testing Tool*

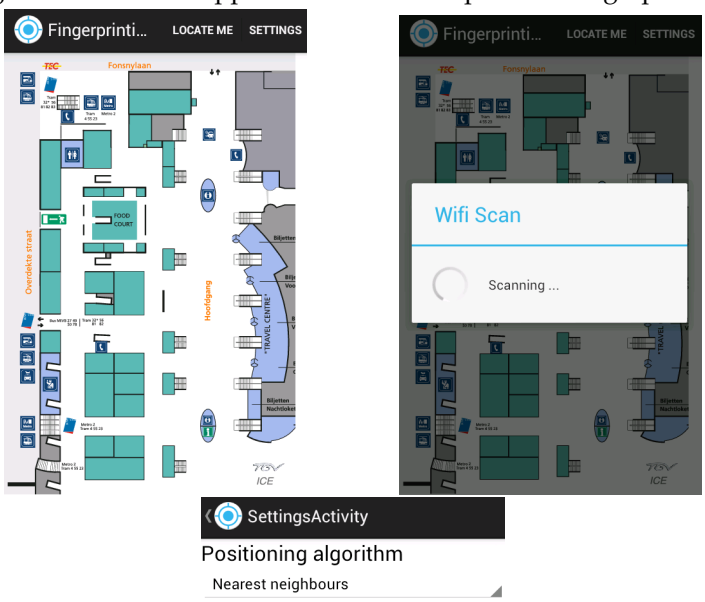
Besides the Android application, a tool was also developed to allow for batch testing of the algorithms. The interface of this tool consists of several tabs, each providing a separate functionality for testing.

The basic idea of this tool is that we have a set of measurements of which we also know the location where they were captured. We can then run the algorithms in various configurations on these measurements. The accuracy of an algorithm can be calculated by looking at the difference between the actual position where the measurement was taken and the one that was estimated by the algorithm.

Other functionalities of the tool include the ability to easily modify the data store with different data sets and to view graphs about the data sets and the performance of the algorithms.

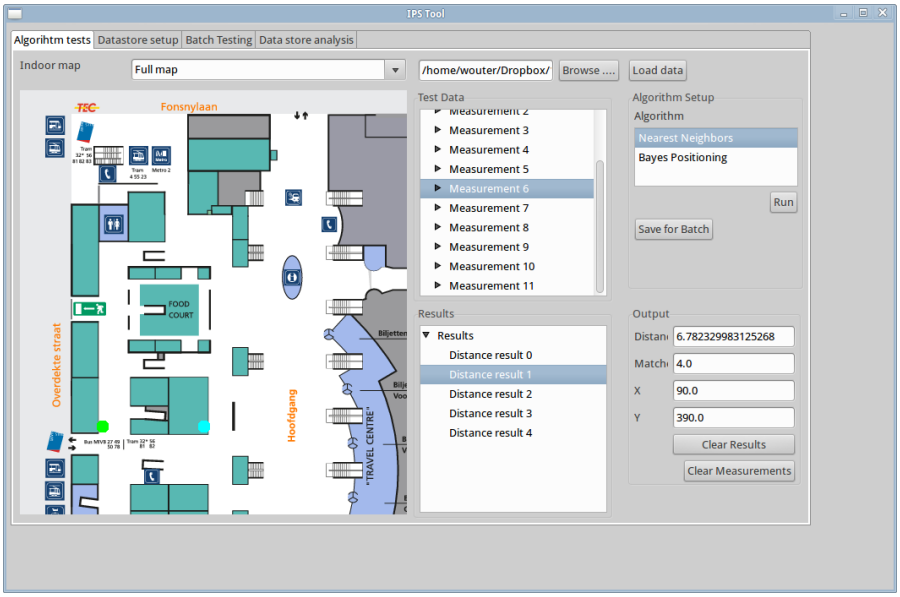
ALGORITHM TEST The first tab shown in Figure 23 is used to test the algorithms on the aforementioned test measurements.

Figure 22: Android application for online phase of fingerprinting



First, a user chooses a folder where the samples (serialised XML fingerprints) are located that they wants to test. If the folder is selected, the fingerprints from this folder are loaded and can be inspected individually in the Test Data panel. The user can then select one or more measurements and an algorithm from the list. A positioning request will then be sent to the server where the specified algorithm will be run for the selected measurements, after which the result will be displayed in the Results panel.

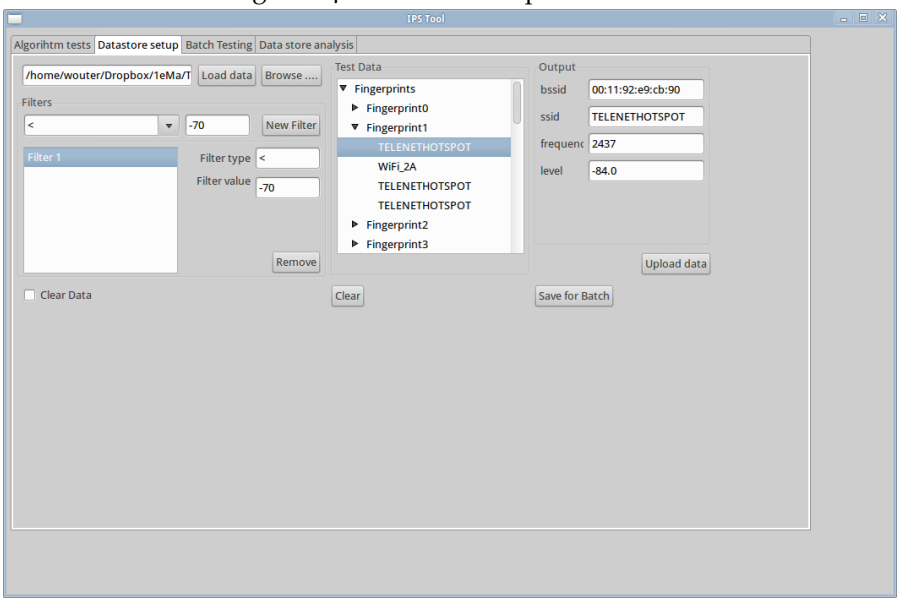
Figure 23: Algorithm tests in the tool



DATA STORE SETUP The next tab in the interface adds support for experimenting with the data in the data store itself. A user can again browse for a folder containing fingerprints that he or she wants to persist to the data store. These can be added to the data that is currently already or the data store can be completely removed first, which allows for easy testing of completely different data sets. There is also the possibility of using filters, which remove all access point power levels in a fingerprint that match the filter. An example would be to filter all the power levels of < -80 dBm since they might be too weak to give reliable results.

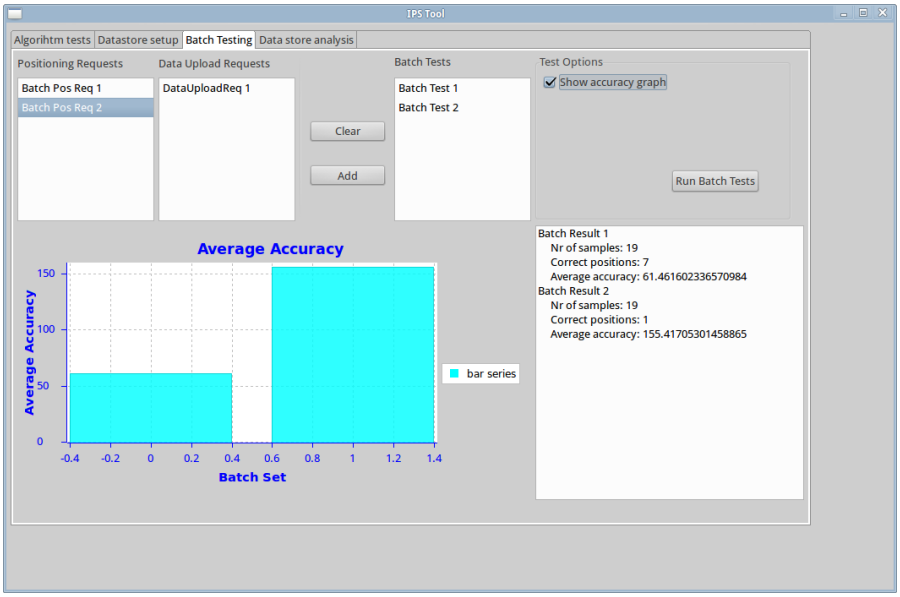
BATCH RUN The two previous tabs also have the possibility, through the `Save for batch` button, to save their execution and run them in batch. In the `Batch Testing` tab we can then see both the available positioning requests as defined in the first tab and the data upload requests that were created in the second tab. A user can first select a positioning request and optionally combine this with a data upload request and add them to the batch tests. This allows the user to mix and match different algorithm configurations on various data sets,

Figure 24: Datastore setup in the tool



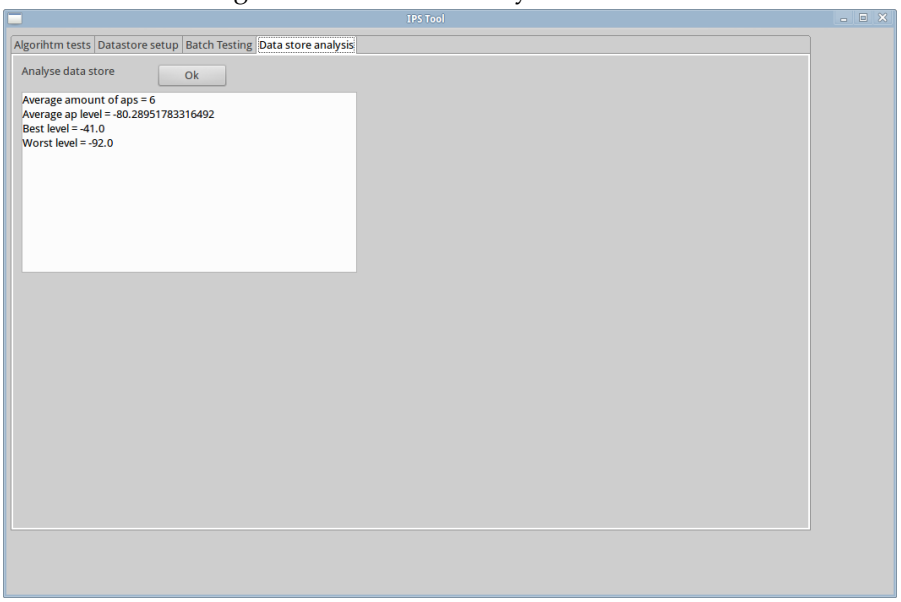
and run them one after the other. The results from this runs can then be compiled and displayed in graphs.

Figure 25: Batch runs in the tool



DATA STORE ANALYSIS The last tab provides some functionality to get information about the fingerprints in the data store such as how many access point there are on average in the fingerprints, what the average power level is among all the fingerprints, and what the worst and best power level is that has been captured.

Figure 26: Data store analysis in the tool



4.4 CONCLUSION

In this chapter, we have seen how the framework can be used to implement an IPS by extending some of the data entities and implementing an algorithm. We also took a look at how we can create client applications that use the framework to collect samples, do positioning and allow for testing of the implemented system. In the next chapter we will see how the implemented system and the applications were used to test it.

5 TESTING THE FRAMEWORK

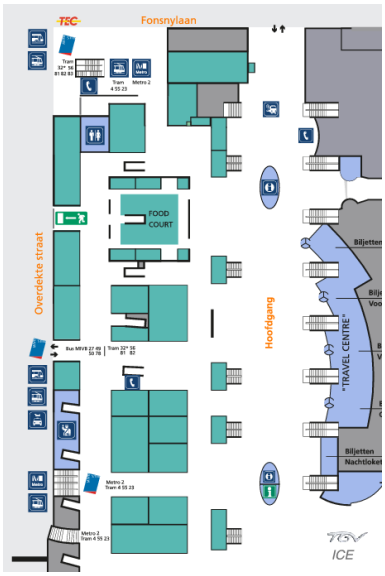
In this chapter, we discuss the testing setups for our indoor positioning system. An indoor location was chosen to test the performance of the IPS by changing different parameters in the testing setup and comparing the results.

5.1 TESTING SETUP

In order to test the WLAN fingerprinting IPS that we implemented by extending our framework, we need to choose a location where we gather the fingerprints and can test the system. So we begin by first giving a general overview of the testing setup: what location was chosen and how the samples were gathered.

INDOOR LOCATION The location that was chosen to test our IPS was the Brussels–South railway station, the biggest railway station in Brussels. We also decided to focus on two parts of the station: the first part of the large hallway that is reached when entering from the main entrance and the second being the food court and its surrounding shops. Figure 27 shows the full area that was used during testing.

Figure 27: Brussels–South railway station



SAMPLE COLLECTING We divide this map in different grids that can vary in size depending on the setup. The samples are then captured by manually traversing the map and using the Android application to do a Wi–Fi scan on that grid position, of which the results are then serialised to an XML file. When doing the sampling process on the whole map, we also collect some extra samples on random grid positions that will be kept separate for testing purposes. The other samples will afterwards be added to the database. This process is then repeated a number of times in order to get a sufficient amount of samples.

LIMITATIONS There are some limitations on the testing because of how the samples are collected. The samples are all collected by one device, and with which the test samples are also collected. This might skew the results, because in the real world the system would be used with different devices with their own hardware which may produce differences in the measurements of the signals. Radio signals are also influenced by the people inside a building, which can vary for the time of day in a station. Therefore the samples were collected at different times to get a more even picture.

5.2 TEST CASES

The tests that were performed can be divided into two types: tests that analyse the performance of the algorithms when changing some parameters and tests that evaluate the algorithms if other training samples sets are used in the data store.

Algorithm Tests

The separate samples that were collected during the sampling phase will be used in a number of ways to test the implemented algorithms.

ACCURACY TESTS Because we know on what grid location we originally collected the test sample, we can use this to test the accuracy of the algorithm that is being used. If we say that our test sample s is a grid position with coordinates (x, y) being the center of the grid and the same for our approximated position p, we can use the Euclidian distance to calculate the distance between the actual position and the position calculated by the algorithm from the sample:

$$d(\mathbf{s}, \mathbf{p}) = \sqrt{\sum_{i=1}^{n} (p_i - s_i)^2}$$

This distance can then be used to denote the accuracy between the actual and the estimated position.

ALGORITHMS Other tests that will be done is to compare the algorithms next to each other by using the aforementioned testing method. By running the algorithms with the same measurements we can see how they perform on specific measurements and by relating all the results we can also see how they perform in general. Comparing the overall performance of the algorithms can then be done.

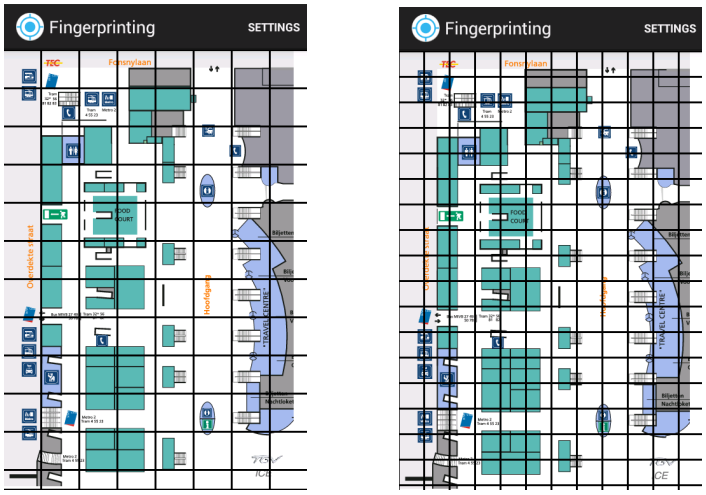
Data Impact Tests

The impact of different data sets on the algorithms can also be used.

GRID SIZES Because our map can be divided into grid cells we can view what the impact will be of different grid cells sizes. Specifically, we would like to see how small we can make the grid sizes while still getting good results. This can be an issue due the fluctuating Wi–Fi signal strengths that make it difficult to differentiate between positions that are too close to each other.

We propose to use three different grid sizes: grids of 10x10 m, 5x5 m and 2.5x2.5 m. Figure 28 shows an example of two different grid sizes.

Figure 28: Two different types of grid sizes



SIGNIFICANT POWER LEVELS When we gather the samples in the offline phase, we currently use all the access point power levels that are returned from the Wi–Fi scan. If an access point has a low signal level it means that the access point is far away from the position and is therefore less reliable than nearby access points. We would therefore need to test at what signal level an access point might become irrelevant or can often induce errors. This can be done by using the filters when uploading the samples.

Samples were collected in different parts of the station. First we have the largest size which comprises a large part of the station and for which we have grid sizes of around 10 by 10 meters. Next, we have two medium size maps where grid sizes of around 5 by 5 meters are used. The location for these two in the station are the food court with its surrounding shops and the main entrance. For both of these locations, fingerprints are also collected with more detailed maps that have grid sizes of around 2.5 by 2.5 meters. A summary of the amount of samples that were collected for each map can be seen in Table 5 and the actual maps can be found in Appendix B.

	GRID SIZE	# SAMPLES	# TEST DATA
Full	10x10m	223	19
Medium 1	5x5m	107	16
Medium 2	5x5m	95	13
Small 1	2.5x2.5m	83	12
Small 2	2.5mx2.5m	141	12

Table 5: Sample amount for the different maps

From the samples that were collected on these areas we can also get some information about the access points on those locations. In Table 6, a summary can be seen with the average amount of access points found on that part of the map, the average power level that was measured from these access point and the best and worst power levels measured from any access point on a position.

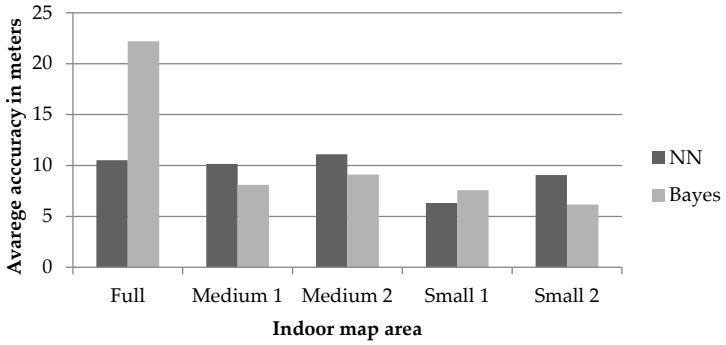
	AVG. # APS	AVG. RSSI	BEST RSSI	WORST RSSI
Full	6	-80.29	-41	-92
Medium 1	4	-78.76	-47	-92
Medium 2	4	-79.58	-49	-93
Small 1	3	-75.73	-45	-92
Small 2	3	-78.11	-46.25	-91

Table 6: Sample amount for the different maps

Grid sizes and algorithm performance

Figure 29 shows the compiled results of the accuracy tests that were done for the algorithms. For each separate map area, we tested both algorithms by using the batch functionality in the tool. This functionality of the tool allows us to calculate the average accuracy that is achieved by the algorithms on the test data. This was then compiled in a bar chart with for each map, two bars representing the average of the nearest neighbour and the bayesian algorithm. Lower values are better, since that means that, on average, the algorithm was more accurate on that map area.

Figure 29: Accuracy of the two algorithms



From these results we can see that we get a higher accuracy when we using smaller grid sizes. We can also see that the nearest neighbour algorithm performs much better than the bayesian on the full map, but doesn't improve as much between the large and the smaller grid sizes as the bayesian method.

Significant power levels

The IPS tool allows us to specify filters when uploading data, which are useful to filter out some unwanted values. This functionality was used to test how the algorithms change when some specific power levels are filtered out because they might be noisy. This was done for both algorithms and for each map area. The results were then compiled in a bar chart with 4 bars for each map area, each representing the accuracy of the algorithm with those filters and one without any filters. Three filters were used to filter out power levels of access points that are smaller than -80, -70 or -60 dBm.

From these results we can conclude that filtering out some of the lower power levels did not have a positive impact on the accuracy. This is because, by using the filters, the average amount of access

Figure 30: Accuracy of nn algorithm with different filters

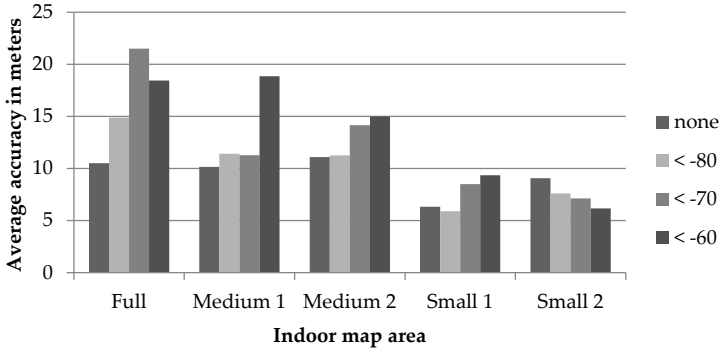
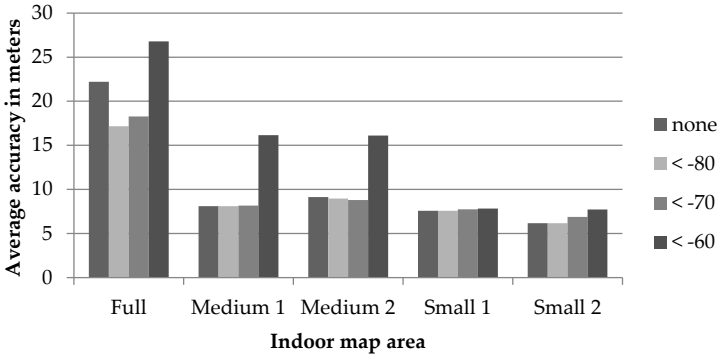
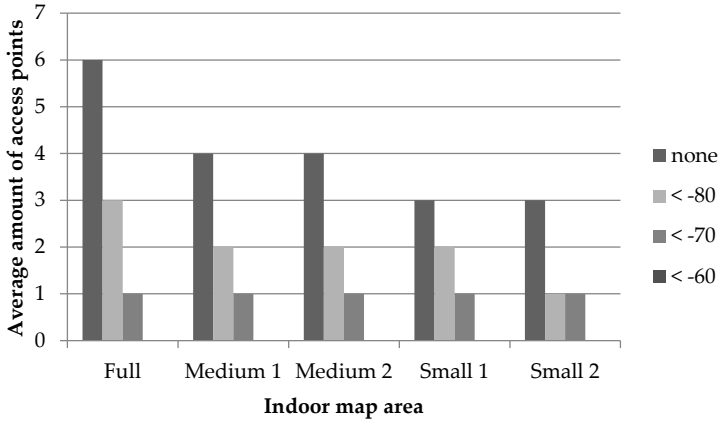


Figure 31: Accuracy of bayesian algorithm with different filters



points in a fingerprints also declines which in turn makes it harder for the algorithm to calculate a correct position. The impact of the filters on the average amount of access point can be seen in Figure 32. We can however see that by removing some of the low power levels (for example smaller than -80 dBm) for some of the maps, the accuracy can be improved. This is because, in these cases, we still have a reasonable amount of access points for the fingerprints.

Figure 32: Average amount of access points with filters



5.5 CONCLUSION

In this chapter we have seen how the framework can be used in practice by using some of the tools that we developed. We saw how fingerprinting can be performed at an indoor location by employing the Android applications. After that we gave some examples of what kind of tests can be performed to test the different algorithms of the system.

From these testing results we can conclude that doing fingerprinting with smaller grid sizes has a positive effect on the accuracy. The lowest grid sizes that we tested were of 2.5 m by 2.5 m and gave accuracy results of around 6 m. This is a mediocre result compared to other projects, but might be improved by using even smaller grid sizes and collecting more samples. We also saw that removing some of the lower value power levels did not always have a positive effect on the accuracy, since this removes certain access points from fingerprints and makes it harder for the algorithms to calculate the position estimate.

6 CONCLUSIONS AND FUTURE WORK

CONCLUSION

In this thesis, we started by giving an overview of the theoretical aspects of radiolocation which is used by indoor positioning systems to do localisation. We discussed how different properties of radio signals, such as signal strength or angle of arrival, can be used together with techniques, such as for instance triangulation, to do indoor localisation. Examples of various systems, using technologies like WLAN or Bluetooth, were presented to explain how we could implement such an IPS. We found that, although numerous systems exist, they were created to test some specific technique and using particular hardware. We therefore also looked at what frameworks already exist, but found that they had a lot of limitations as well. Most were found to be out–dated or not extensible enough.

For these reasons we created a new extensible framework for indoor positioning, which aims at modern mobile devices. We also proposed some goals that our framework would need to accomplish:

- Our framework needs to support as many techniques as possible by providing basic building blocks that can be reused and expanded for other systems.
- Since data capturing is an expensive task, data measurements need to be in a standardised format so that they can be reused and are not be directly coupled to the framework and the data store.
- We want to have a flexible system where the algorithms should be as interchangeable as possible.

We first discussed this framework in a top–down approach, beginning with a high level overview of the data flow and architecture of the framework. After that we continued by providing details on the design of the framework, such as specify the data entities and the design of the algorithms and server architecture. Some implementation aspects are also handled since they had an impact on the framework.

How the framework can be used was then explained by providing an example of how we extended the framework to implement an IPS using a WLAN scene analysis method. Modifications to the data entities were done to support the technology, as well as the implementation of two different algorithms: a nearest neighbours algorithm and a statistical method. The IPS was made to work on Android mobile

devices and applications were made for the platform to capture data and do positioning, as well as another Java tool that uses the framework to allow for easy testing of the system.

The different tests that that were done using that tool were then described, followed by the results that were achieved. And although the tests showed that we only achieved a mediocre accuracy, it proved that a simple IPS can easily be set up by using the framework and that the goals of the framework were met:

- The framework can support many techniques without having to do a lot of drastic changes. The basic building blocks that are available can be easily reused and expanded for other systems.
- The captured data is in a standardised format and can be easily reused.
- New algorithms can easily be added and are interchangeable.

FUTURE WORK

Some methods on how to improve the framework and our indoor positioning system still exist, but were not pursued in this thesis. We can divide these in two types: methods that try to improve the offline stage of fingerprinting by minimising the amount of data that needs to be collected during the offline phase, and methods that improve the online stage of fingerprinting by ameliorating the algorithms.

Improving the Offline Stage

When creating our initial fingerprint database, a rather big manual effort has to be done in order to acquire the samples. Nevertheless, some techniques can be used to reduce the amount of work that needs to be done.

GENERATING SAMPLES Li et al. [13] explain that in the conventional method of generating the database we do not utilise the spatial correlation of measurements sampled at adjacent reference points. This implies that we can use generate a denser database using only a selected number of reference points and therefore reduce some of the labour and time needed in the offline stage. The method we can us for this is called *interpolation*. Two possible techniques for interpolation are: *kriging* and *inverse distance weighting*.

Kriging is a technique to *"interpolate the value of a random field (e.g., the elevation, z, of the landscape as a function of the geographic location) at an unobserved location from observations of its value at nearby locations"* [25]. It is a type of linear least squares estimation algorithm that aims to estimate the value of an unknown real–valued function,

f, at a point, x^* given the values of the function at some other points $x_1, x_2, ..., x_n$.

Inverse distance weighting is also a technique that assigns values to unknown points by using values from usually scattered set of known points. The value of the unknown point is the weighted sum of the value of N known points [24].

Improving the Online Stage

The online stage of a fingerprinting approach can also be improved by using extra information with the algorithms or combining different algorithms.

MOTION MODEL One improvement that can be done is by using a *motion model* as described in Section 4.2.2. We would then take into account the previous locations of the user — from previous position estimates — and modelling how far they user could move between positioning requests, we could remove some of the noisy outliers.

AUTOMATIC CONFIGURATION Another possibility that could be explored is to see how different algorithms perform on different parts of the indoor location. It could be that one algorithm might achieve better results when fewer base stations can be seen, and therefore automatically configure positioning requests depending on the observed measurements.

COMBINATIONS The last improvement that we propose is to explore the performance of running different algorithms in parallel for a single positioning request. Better results might be achieved by looking at the results of the different algorithms and, based on this, pick a position estimate to return. This could be the average position, by using some kind majority voting between the algorithms or assigning some weights to each algorithm.

A

XML EXAMPLES

```
<WLANFingerprint>
   <position class="ips.data.entities.wlan.GridPosition" id="0">
      <map class="ips.data.entities.wlan.GridMap">
         <mapFileName></mapFileName>
         <mapId>full</mapId>
         <gridSize>70</gridSize>
      </map>
      <y>690</y>
      <x>330</x>
      <bottomY>720</bottomY>
      <leftX>300</leftX>
      <rightX>360</rightX>
      <topY>660</topY>
   </position>
   <wlanMeasurement>
      <apMeasurements class="java.util.HashMap">
         <wlanMeasurement>
            <accessPoint id="00:1d:6a:a6:fe:35">
               <signalProperties class="java.util.ArrayList">
                  <signalProperty>RSSI</signalProperty>
               </signalProperties>
               <BSSID>00:1d:6a:a6:fe:35</BSSID>
               <SSID>bbox2-f848</SSID>
               <capabilities>[WPA2-PSK-CCMP][ESS]</capabilities>
               <frequency>2412</frequency>
            </accessPoint>
            <powerLevels>
               <signalMeasurements class="java.util.HashMap">
                  <signalMeasurement>
                     <singalProperty>RSSI</singalProperty>
                     <measurements>
                        <values class="java.util.ArrayList">
                           <double>-82.0</double>
                        </values>
                     </measurements>
                  </signalMeasurement>
               </signalMeasurements>
               <average>-82.0</average>
            </powerLevels>
         </wlanMeasurement>
         <wlanMeasurement>
            <accessPoint id="00:24:a5:b4:81:ad">
               <signalProperties class="java.util.ArrayList">
                  <signalProperty>RSSI</signalProperty>
               </signalProperties>
               <BSSID>00:24:a5:b4:81:ad</BSSID>
```

```
            <SSID>AXIS</SSID>
            <capabilities>[WPA-PSK-TKIP][WPS][ESS]</
                capabilities>
            <frequency>2437</frequency>
         </accessPoint>
         <powerLevels>
            <signalMeasurements class="java.util.HashMap">
               <signalMeasurement>
                  <singalProperty>RSSI</singalProperty>
                  <measurements>
                     <values class="java.util.ArrayList">
                        <double>-84.0</double>
                     </values>
                  </measurements>
               </signalMeasurement>
            </signalMeasurements>
            <average>-84.0</average>
         </powerLevels>
      </wlanMeasurement>
      <wlanMeasurement>
         <accessPoint id="00:11:92:e9:fa:b0">
            <signalProperties class="java.util.ArrayList">
               <signalProperty>RSSI</signalProperty>
            </signalProperties>
            <BSSID>00:11:92:e9:fa:b0</BSSID>
            <SSID>TELENETHOTSPOT</SSID>
            <capabilities>[ESS]</capabilities>
            <frequency>2462</frequency>
         </accessPoint>
         <powerLevels>
            <signalMeasurements class="java.util.HashMap">
               <signalMeasurement>
                  <singalProperty>RSSI</singalProperty>
                  <measurements>
                     <values class="java.util.ArrayList">
                        <double>-79.0</double>
                     </values>
                  </measurements>
               </signalMeasurement>
            </signalMeasurements>
            <average>-79.0</average>
         </powerLevels>
      </wlanMeasurement>
      <wlanMeasurement>
         <accessPoint id="00:23:f8:2c:1c:18">
            <signalProperties class="java.util.ArrayList">
               <signalProperty>RSSI</signalProperty>
            </signalProperties>
            <BSSID>00:23:f8:2c:1c:18</BSSID>
            <SSID>Free Sam&apos;s Cafe</SSID>
            <capabilities>[ESS]</capabilities>
            <frequency>2422</frequency>
```

```
      </accessPoint>
      <powerLevels>
         <signalMeasurements class="java.util.HashMap">
            <signalMeasurement>
               <singalProperty>RSSI</singalProperty>
               <measurements>
                  <values class="java.util.ArrayList">
                     <double>-79.0</double>
                  </values>
               </measurements>
            </signalMeasurement>
         </signalMeasurements>
         <average>-79.0</average>
      </powerLevels>
   </wlanMeasurement>
   <wlanMeasurement>
      <accessPoint id="00:17:94:fe:0f:40">
         <signalProperties class="java.util.ArrayList">
            <signalProperty>RSSI</signalProperty>
         </signalProperties>
         <BSSID>00:17:94:fe:0f:40</BSSID>
         <SSID>TELENETHOTSPOT</SSID>
         <capabilities>[ESS]</capabilities>
         <frequency>2412</frequency>
      </accessPoint>
      <powerLevels>
         <signalMeasurements class="java.util.HashMap">
            <signalMeasurement>
               <singalProperty>RSSI</singalProperty>
               <measurements>
                  <values class="java.util.ArrayList">
                     <double>-82.0</double>
                  </values>
               </measurements>
            </signalMeasurement>
         </signalMeasurements>
         <average>-82.0</average>
      </powerLevels>
   </wlanMeasurement>
   <wlanMeasurement>
      <accessPoint id="00:11:92:e9:cb:90">
         <signalProperties class="java.util.ArrayList">
            <signalProperty>RSSI</signalProperty>
         </signalProperties>
         <BSSID>00:11:92:e9:cb:90</BSSID>
         <SSID>TELENETHOTSPOT</SSID>
         <capabilities>[ESS]</capabilities>
         <frequency>2437</frequency>
      </accessPoint>
      <powerLevels>
         <signalMeasurements class="java.util.HashMap">
            <signalMeasurement>
```

```
                    <singalProperty>RSSI</singalProperty>
                    <measurements>
                       <values class="java.util.ArrayList">
                          <double>-63.0</double>
                       </values>
                    </measurements>
                 </signalMeasurement>
              </signalMeasurements>
              <average>-63.0</average>
           </powerLevels>
        </wlanMeasurement>
        <wlanMeasurement>
           <accessPoint id="00:11:92:e9:d4:00">
              <signalProperties class="java.util.ArrayList">
                 <signalProperty>RSSI</signalProperty>
              </signalProperties>
              <BSSID>00:11:92:e9:d4:00</BSSID>
              <SSID>TELENETHOTSPOT</SSID>
              <capabilities>[ESS]</capabilities>
              <frequency>2462</frequency>
           </accessPoint>
           <powerLevels>
              <signalMeasurements class="java.util.HashMap">
                 <signalMeasurement>
                    <singalProperty>RSSI</singalProperty>
                    <measurements>
                       <values class="java.util.ArrayList">
                          <double>-76.0</double>
                       </values>
                    </measurements>
                 </signalMeasurement>
              </signalMeasurements>
              <average>-76.0</average>
           </powerLevels>
        </wlanMeasurement>
     </apMeasurements>
  </wlanMeasurement>
</WLANFingerprint>
```

Figure 33: Full and other map areas.

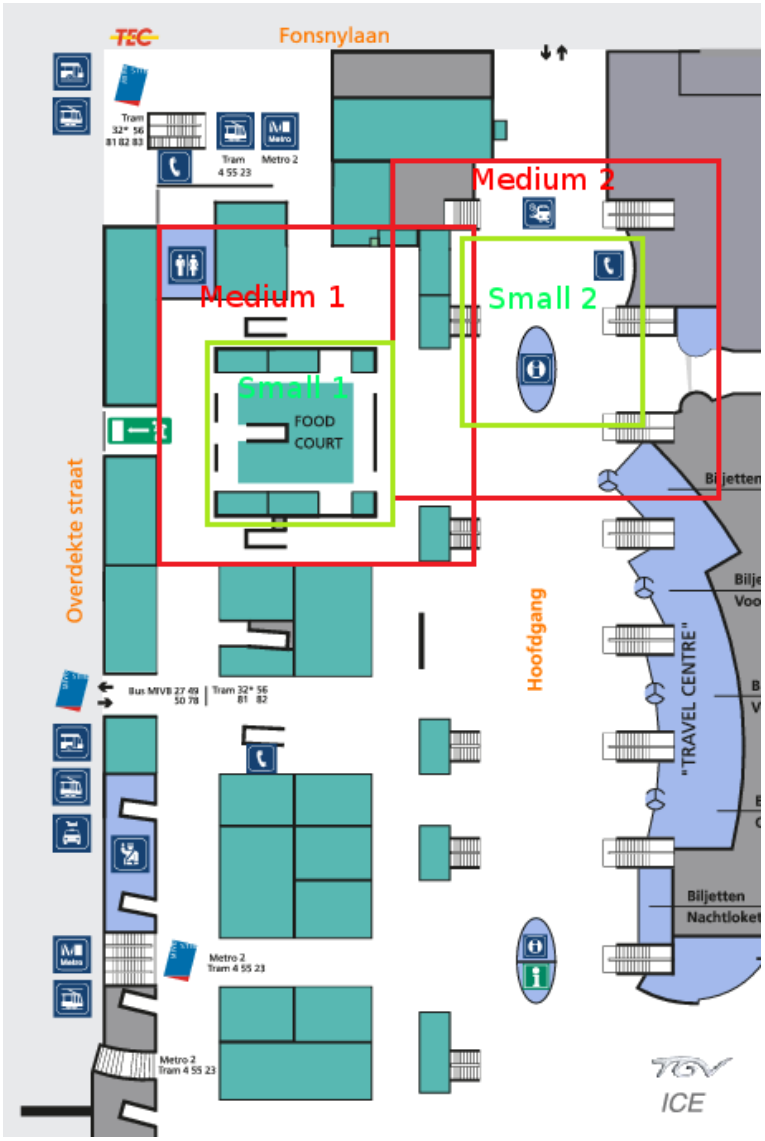


Figure 34: Medium 1 & 2.

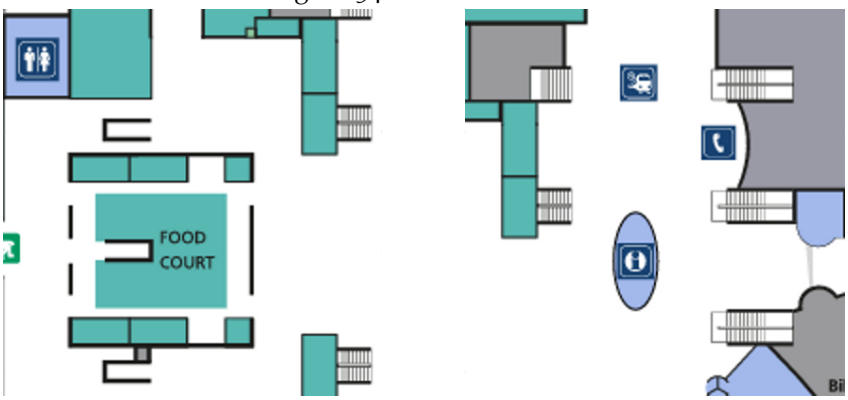
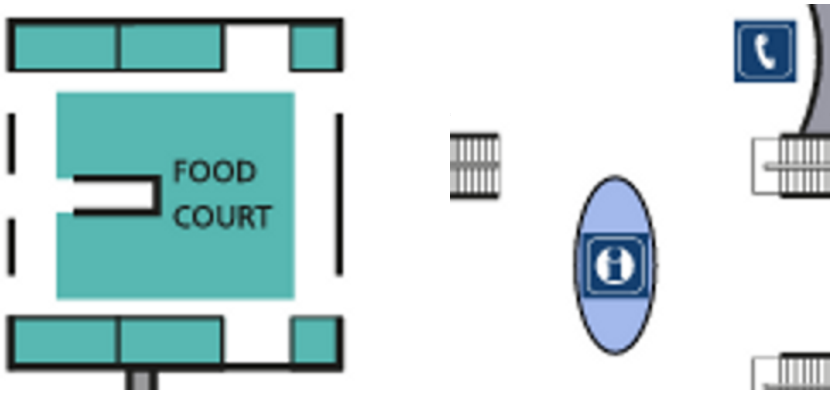


Figure 35: Small 1 & 2.



BIBLIOGRAPHY

[1] P. Bahl. RADAR: An in-building RF-based user location and tracking system. 2000.

[2] R Battiti and NT Le. Location-aware computing: a neural network model for determining location in wireless LANs. 2002.

[3] Inc. Bluetooth SIG. A look at the basics of Bluetooth wireless technology, 2012.

[4] JJ Caffery. Overview of radiolocation in CDMA cellular systems. *Communications Magazine*, (April):38–45, 1998.

[5] Kevin Curran, Eoghan Furey, Tom Lunney, Jose Santos, Derek Woods, and A. McCaughey. An evaluation of indoor location determination technologies. *Journal of Location Based Services*, 5(2):61–78, 2011.

[6] Silke Feldmann and Kyandoghere Kyamakya. An indoor Bluetooth-based positioning system: concept, implementation and experimental evaluation. *International Conference on Wireless Networks*, pages 109–113, 2003.

[7] Gunter Fischer and Burkhart Dietrich. Bluetooth indoor localization system. *Proceeding Workshop on Positioning*, pages 147–56, 2004.

[8] F Forno and G Malnati. Design and implementation of a Bluetooth ad hoc network for indoor positioning. 2005.

[9] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional, 1994.

[10] Dan Grossman. A sophomoric introduction to shared-memory parallelism and concurrency. 2012.

[11] Martin Kodde. Bluetooth communication and positioning for location based services. 2005.

[12] C Laoudias. The airplace indoor positioning platform for android smartphones. 2012.

[13] Binghao Li, James Salter, and AG Dempster. Indoor positioning techniques based on wireless LAN. 2006.

[14] Stephan Linzner and Daniel Kersting. Smart Space - an indoor positioning framework. 2009.

[15] H. Liu, Houshang Darabi, Pat Banerjee, and Jing Liu. Survey of wireless indoor positioning techniques and systems. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 37(6):1067–1080, 2007.

[16] G Machovec. *Telecommunications, networking, and Internet glossary*. LITA monographs. Library and Information Technology Association, 1993.

[17] Marius Marcu, B Stratulat, Iulia Stratulat, and Anania Girban. A low power framework for WLAN indoor positioning system. *Proceedings of the 13th WSEAS International Conference on Computers*, pages 394–399, 2009.

[18] Lionel M. Ni, Yunhao Liu, Yiu Cho Lau, and Abhishek P. Patil. LANDMARC: indoor location sensing using active RFID. *Wireless Networks*, 10(6):701–710, November 2004.

[19] C. Enrique Ortiz. An introduction to near-field communication and the contactless communication API, 2008.

[20] Veljo Otsason, Alex Varshavsky, and A LaMarca. Accurate gsm indoor localization. *UbiComp 2005*, (iv):141–158, 2005.

[21] Busra Ozdenizci, Kerem Ok, and Vedat Coskun. Development of an indoor navigation system using NFC technology. *Fourth International Conference on Information and Computing Science*, (April):25–27, 2011.

[22] Guenther Retscher, Eva Moser, and Dennis Vredeveld. Performance and accuracy test of the WLAN indoor positioning system ipos. *Workshop on Positioning, Navigation and Communication*, pages 7–16, 2006.

[23] Suzanne Ross. Wherever you go, there is connectivity, 2001.

[24] Jerome Sacks, William J Welch, Toby J Mitchell, and Henry P Wynn. Design and analysis of computer experiments. *Statistical Science*, 4(4):409–423, 1989.

[25] Donald Shepard. A two-dimensional interpolation function for irregularly-spaced data. In *Proceedings of the 1968 23rd ACM national conference*, pages 517–524. ACM Press, 1968.

[26] Reid Simmons and Sven Koenig. Probabibstic robot navigation in partially observable environments. 1994.

[27] Silverio C. Spinella, Antonio Iera, and Antonella Molinaro. On potentials and limitations of a hybrid WLAN-RFID indoor positioning technique. *International Journal of Navigation and Observation*, 2010:1–11, 2010.

[28] Kha Tran, Dinh Phung, and Brett Adams. Indoor location prediction using multiple wireless received signal strengths. *AusDM 2008 : Proceedings of the 7th Australasian Data Mining Conference*, 2008.

[29] David Tse and Pramod Viswanath. *Fundamentals of Wireless Communication*. Cambridge University Press, 2005.

[30] Moustafa Youssef and Ashok Agrawala. The Horus location determination system. *Wireless Networks*, 14(3):357–374, January 2007.