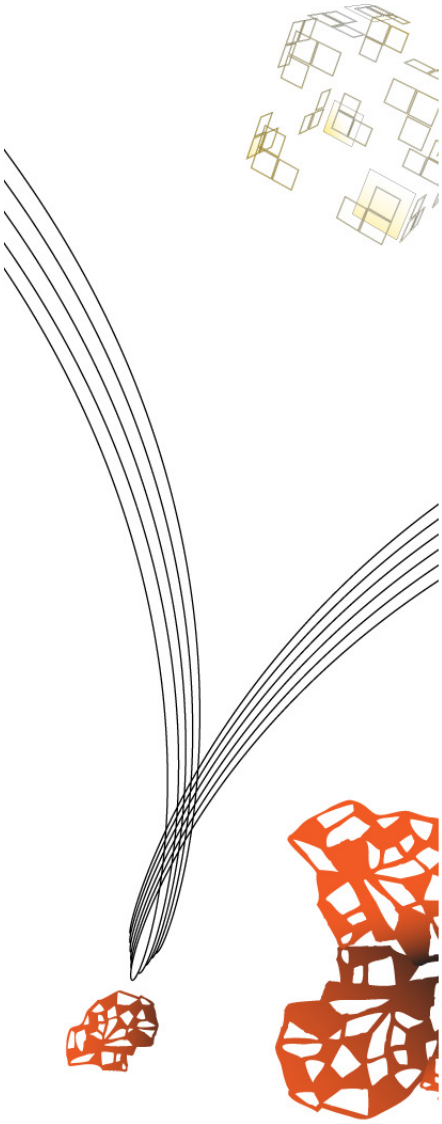
Opportunistic sensing & Aggregation network Using Smartphones and Sensor nodes



Author:
Dennis Heuven
(s1280627)

Master Thesis

Supervisor:
Ir J. Scholten

Committee:
Ir J. Scholten
prof. dr. ing. P.J.M. Havinga
ir. O. Türkes

Chair:
Pervasive Systems Group

30-07-2014
Version 1.0
Pages: 72

UNIVERSITEIT TWENTE.

This research is part of the COMMIT/SenSafety project
http://commit-nl.nl

Revision Table

Version	Date	Changes
0.1	17-05-2014	First working version (WIP)
0.2	22-05-2014	Minor changes and additions
0.3	08-06-2014	More content
0.4	11-06-2014	More content
0.5	21-06-2014	Further writing and improvements
0.6	28-06-2014	Further improvements
0.7	15-07-2014	Minor corrections
1.0	30-07-2014	First final version

Contents

List of Figures

List of Tables

Abstract

The COMMIT/SenSafety project focuses on using heterogeneous sensor networks for safety and (perceived) security. Part of this safety is sensing the environment. Classically sensing the environment is done using a static network of sensor nodes spread in the area of interest however this has some disadvantages such as monitoring only fixed positions and the requirement that nodes are connected with one another in order to upload their measurements. It is desired that a whole city could be monitored with only a limited amount of nodes which should have dynamic locations. A viable (partial) replacement for traditional sensor nodes are smartphones. Smartphones can act as the communication link between separate sensor nodes thereby increasing the possible size of the network as nodes do not have to be in communication range of one another anymore. Smartphones are mobile as their owners carry them around, they are often recharged and have multiple ways for communication. What are the challenges in using smartphones in environmental sensing and how would one do so? This thesis attempts to answer these questions by answering the research question
"How can smartphones be used in environmental sensing, in the context of a city-wide project, and what are the challenges to overcome?".

A prototype was designed and implemented to demonstrate the feasibility of using smartphones in environmental sensing with the focus on using opportunistic communications instead of cellular communications. The prototype consists of a webserver, Android smartphones and sensor nodes. The smartphones communicate with other smartphones and with the sensor nodes. The communication to the sensor nodes is achieved using Bluetooth Low Energy, the communication between smartphones is achieved using Cocoon. Cocoon is a technology that is being developed on the University of Twente which uses WiFi- hotspot and client mode to enable opportunistic inter-smartphone communication. The prototype was tested in both a simulator and in a testbed. The simulator was used to simulate larger scale networks and scenarios which could not be tested in the testbed and the testbed was used to demonstrate the real feasibility of usage. During these tests with the testbed some irregularities were observed such as device specific behavior for Bluetooth, Bluetooth Low Energy and WiFi hotspots. The irregularities observed are listed in Appendix A. For the prototype the CC2541 SensorTag was used as a sensor node and either a combination of one HTC one V, two Moto G's and Galaxy S4 mini's were used.

The prototype was able to communicate with the sensor node using Bluetooth Low Energy and to other smartphones using Cocoon although some device-specific problems were observed. The smartphones aggregated and shared the data which were later visualized to the users. The smartphones were able to upload their data to the external server in which the uploaded results are visualized via a website. The Future work is related to having more tests and a larger testbed to improve the quality of testing of the prototype.

1 Introduction

Smartphones are more and more common in everyday life. Nowadays it's possible to use a smartphone for a lot of things like looking up the weather forecast, listening to music or using your smartphone for navigation. The internet of things is an upcoming trend where everyday devices contain sensors and communicate with one another. Another step towards the internet of things can be made by the introduction of small low power detached sensors that extend the sensing capabilities of smartphones. These extensions communicate with the smartphone via Bluetooth, WiFi or USB. Alternately smartphones can be connected to external sensing devices. Sensors incidentally report inaccurate or incorrect data, which can be compensated by sharing the sensor data with nearby nodes that also measure the same characteristics, thereby improving the correctness and overall accuracy of the sensors. The sharing and aggregation of the data can also be performed by the interested party, in this case the smartphones. By letting the smartphones share the data it reduces the energy required by the sensor nodes (no inter-sensor node communication costs). An inter-smartphone communication method is required to share the information between the different smartphones and a communication method is required to get the data from the sensor board to the smartphone. It is also desired to transmit the sensor data to an external server to publicize the measured data and perform big-data analysis.

The COMMIT/SenSafety project focuses on using heterogeneous sensor networks for safety and (perceived) security, in which the environment is an important factor. Aspects that can be measured in the sensor network are, for example, air quality (carbon dioxide levels, presence of chemicals), noise pollution, pressure, humidity, temperature and radiation. This project focuses on the city-wide usage of a network consisting of smartphones and sensor nodes in which volunteers help measuring and reporting the environment. In the project the provided sensor nodes are spread in the city or given to users. Users have smartphones with special application installed to share, collect and view the sensor data. It is assumed that the nodes do not have capabilities to reliably get their exact location and are thus dependent on the smartphones for localization. Furthermore it is assumed that the sensors and smartphones are not connected to the internet during operation (except for the smartphone when uploading the data).

1.1 Objectives

The research question this thesis aims to answer is
"How can smartphones be used in environmental sensing, in the context of a city-wide project, and what are the challenges to overcome?"

The sub-questions to aid in solving the main research question are:

1. *What is opportunistic sensing and what are its challenges?*
2. *What is environmental sensing and what are its challenges?*

3. *What capabilities do smartphones have to offer for environmental sensing, and how feasible are their uses for environmental sensing?*

4. *What are the challenges in inter-smartphone communication in the context of an opportunistic network?*

To demonstrate the real-world feasibility a prototype was developed. The tasks related to the prototype are to:

- Design a communication layer to share (aggregated) sensor data between smartphones;
- Design an aggregation method for the sensor data;
- Design a communication method which uses Bluetooth Low Energy to retrieve sensor data from a sensor board;
- Create a prototype to validate the designs and prove its feasibility. The prototype should visualize the data for the user. The smartphone to smartphone communication method should at least be validated in both a simulator and in a real scenario. If time allows the sensor data should optionally be send towards an external server.

Furthermore there are some additional (technical) requirements for the prototype:

- Sharing of sensor data between smartphones should be realized to increase sensor accuracy
- The sharing of sensor data and normal operation should be possible without requiring infrastructure (no WiFi or cellular internet connection).
- The (aggregated) sensor data should optionally (with the consent of the user) be uploaded to an external server when connected to the Internet.
- Uploading data via a mobile subscription can become costly when uploading without WiFi connection and using a lot of data, thus the uploading should be avoided whilst not connected through WiFi. Furthermore it is not possible to force upload via mobile subscription whilst connected to WiFi on Android devices.

1.2 Context

The context of the final project can be identified as an urban-outdoor-environment with users (smartphone carriers) that move around in the area. The expected (useful) movement speed is walking (5km/h) up to cycling speed (20km/h).
The final project focuses on Android smartphones only as the current smartphone market consists mainly of Android smartphones (78.9% Android vs 15.5% IOS vs 5.6% other) [21].
The expected density of the network (smartphones and sensor nodes in an area) will be varying depending on the number of participants, the location and the time. In several areas of interest the number of nodes is expected to be higher than average.
The time of the sensor data is rated as a bit time sensitive, the measured data

can only be aggregated with other data of the same time. The external upload should accept old aggregated data.
To make sure the data is valid (and thus useful) it is important that the data is not tampered with thus the message integrity is important.
The sensor data is not confidential as everyone could retrieve the same data of the nodes.
The expected data rate of messages is low as sensor nodes only periodically generate new data and the sharing of sensor data occurs periodically.
The sensor data should be aggregated only with related sensor data of nearby nodes as the data is depended on the location.
Authenticity of messages is not very important during the aggregation however the sources are important during the upload as sensor data of corrupted / compromised nodes should be ignored.
The availability of the sensor data is important for the aggregation, however it is not always required to upload all aggregated data as other nodes can potentially upload the same data as the data is shared with nodes participating in the aggregation.

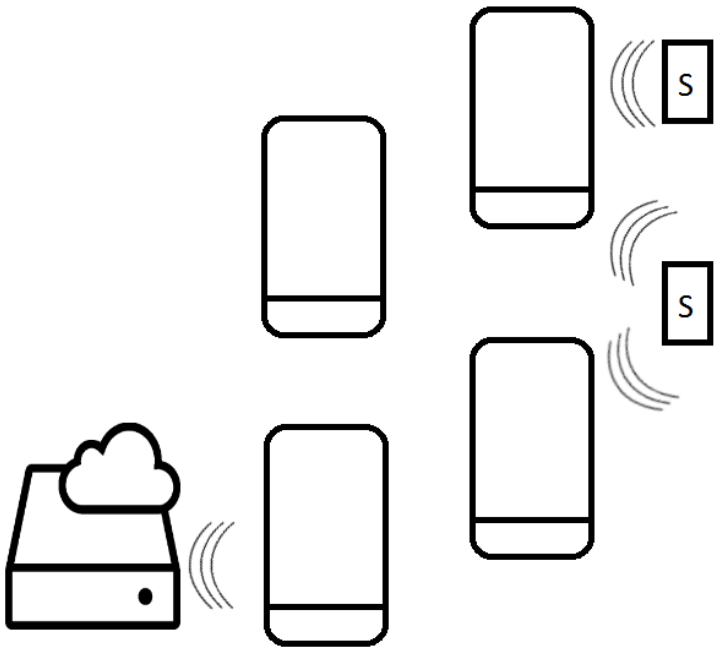


Figure 1: Scenario

1.3 Outline of the thesis

The remainder of this thesis is ordered as follows. Chapter 2 describes the related work on relevant parts of the project. Chapters 3 and 4 cover the sub questions 1 and 2. Chapters 5 and 7 cover the operating system Android and the communication techniques it can use, covering sub question 3 and 4. Chapter 6 provides additional information on the sensor nodes to be used in conjunction

with the smartphones. Chapters 8 and 9 cover the proposed design, which is later validated in chapter 10. Finally chapter 12 features a discussion on the usability, lists the conclusions and suggests future work.

2 Related Work

This section describes relevant works related to parts of the thesis. Most of the relevant work is also cited in the concerned chapters.

Related projects
Several projects were designed with the goal of improving neighborhoods using environmental sensing. Most published projects were open, that is new users can join the project and participate in the sensing using their smartphones and sensors. Often these projects use sensors that are available on the phone itself or as an attached accessory as opposed to connecting to sensor nodes. Other projects used external sensors without sharing the results or sharing whilst requiring an internet connection. The authors of [12] demonstrated an approach to use Bluetooth for the sharing of sensor data between a sensor node and a smartphone, however it was designed only for a single phone with the goal of measuring temperature and humidity. The paper [13] was published by the same author and extended the previous paper by using multiple Bluetooth sensors. The approach however it still uses only one smartphone. The authors of [28] validate the feasibility of using Bluetooth as the communication method in sensor networks, which appears to be usable however there were some complications with for example pairing.

Sharing and aggregation
The sharing & aggregation of data has been done before, albeit using an internet connection. Also most sensor networks perform the aggregation outside the sensor network or incrementally towards the sink of the network. In this thesis the processing is done inside the network. Sharing of data in Android is pretty common for a single phone to a single other device (think of file sharing or wireless headset via Bluetooth) is pretty common however automatically sharing is far less used and sharing in combination with an infra-structureless environment is even rarer, especially sharing with multiple devices at the same time.

Sharing communication techniques
Using Bluetooth Low Energy for data retrieval is not very common yet due to the novelty, and it is used even less with smartphones (in a sensor network). The reason for this is that Bluetooth Low Energy is fairly new especially in smartphones, as the standard was only merged into the Bluetooth standard in 2010. The support for Bluetooth Low Energy in smartphones is even more recent as it was introduced in Android version 4.3+ which was released at 24 Juli 2013.

The usage of Cocoon (Chapter 7.2) is unique for the purpose of sharing sensor and aggregation data as it has not yet been officially published, however a similar technique is starting to be increasingly popular. The related technique for iOS is the "Multipeer connectivity Network" which enables IPhones to directly connect to nearby IPhones using Bluetooth and WiFi in order to share data which can be in the form of messages, information streams or resources like images, movies or documents. The most well known application to use the "Multipeer connectivity

Network" technique of iOS is FireChat. Firechat lets users chat with nearby people, even if there is no phone coverage or internet connection. FireChat has recently been ported to Android [33], thereby directly competing with Cocoon. The FireChat source code (and application) was not available during most of the thesis work so it was not taken into account in the design or prototype. Firechat is available at `https://opengarden.com/firechat`, the Google Play Store or the iOS App Store.

Opportunistic sensing with smartphones
The authors of [17] proposed MetroSense, a network architecture for urban-scale people-centric sensing with design goal of broad application and sensor heterogeneity support. MetroSense uses an opportunistic sensor networking approach to scale to large areas. MetroSense has been designed with regularly recharged sensors in mind like cell phones and PDAs.
Opportunistic sensing and opportunistic networking is often associated with human-centric ubiquitous systems, such as in crowd sourcing and participatory sensing applications such as [18, 22, 25, 34] which focus for example on noise pollution, air quality or are focusing on human activity recognition such as [19] in which the authors propose CrowdSense@Place (CSP) which focuses on characterizing places with opportunistic crowdsensing using smartphones. In CSP smartphones collect audio traces which are compared to a set of known traces to determine in what type of environment the user is. Other projects which use opportunistic sensing but not with smartphones are CarTel [23] which focuses on in-car computers which measure the environment, can be queried to retrieve the current status and can optionally share data with other nodes or to the internet using opportunistic wireless connectivity; The authors of [31] proposes to use opportunistic sensing in train safety systems to make a distinction between carriages from different trains.

Testing
The authors of [36] introduced a simulation environment for smartphone sensor networks that calculates smartphone specific properties of a sensor network. The provided simulation environment was poorly documented and it was unclear how to use it thus is was not used for the validation.

3 Environmental sensing

Environmental sensing is the sensing of one's environment, often using sensor networks. There are several fields of environmental monitoring, such as air quality monitoring, water quality monitoring, soil quality monitoring and so on, each field focusing on measuring and interpreting different environmental parameters. The environmental sensing in this thesis is focused on measuring an urban environment. Some characteristics that can be monitored in an urban environment are, for example, the humidity, temperature, noise (pollution), pressure or radiation of an area.

In environmental sensing it is important to have either very accurate measurements or a lot of correlating measurements, depending on the proffered accuracy and the area of interest. Sensor networks typically consist of a lot of resource constrained devices which often have a very limited power supply. Al tough sensor nodes are typically relatively cheap it is not desired to replace the sensor nodes after they have been deployed, thus the lifetime of sensor nodes should be as long as possible.

Environmental sensing is broad term and "environmental sensing networks" can be disseminated by some parameters:

- Where is the network deployed: Indoors/Outdoors/Oceans/In the air?
- What should be measured?
- What is the area of interest?
- What should be done with the data?

The challenges in environmental sensing are often directly related to the field of the monitoring, but there are also some common challenges.The common challenges for environmental sensing can be divided into two major categories: challenges related to the collection of data and the processing (and retrieving useful data) of the collected data. The challenges for retrieving data in the context of sensor nodes is further explained in chapter 6.

The common challenges for the processing of the collected data are:

1. Correlation of data that is collected, in this thesis the challenges related to the locations of the data.
2. Information interpretation and extraction, what does the data represent and which parts are useful or special.
3. Privacy related challenges, removing the user-details whilst keeping the data useful.
4. Security and authenticity of the data, preventing attacks (such as data injection) on the system, whilst preserving privacy.

4 Opportunistic sensing

Opportunistic systems consists of changing groups of nodes which temporary work and communicate together to archive a common goal together, in the setting of this thesis they work together to perform data aggregation and sharing of their aggregated data. Opportunistic networks are typically able to work in the absence of a stable and permanent communication network, such as WiFi or a cellular network thus being able to spreading information in an area typically not available by other systems due to coverage problems (either case of non-existing infrastructure, overloaded infrastructure or disabled infrastructure). In participatory sensing there is typically no single data producer of data as the data is a collection of data provided by different nodes. There are also no guarantees to the availability of data as providers participate voluntary and can stop the participation at any time. The nodes in a opportunistic network are typically mobile and can cover large areas over time, which is the case in this thesis as the nodes used are smartphones which are carried by participants.

Important challenges in opportunistic sensing are

- *The participation of users*
  In order to let groups of nodes work together you need nodes in the first place. In this scenario the nodes are smartphones which are carried by the participants which participate voluntary. Because of the voluntary participation users can stop participation at any time, thereby reducing the availability. To increase the number of participants it is important to know the motivations for initial and continued participation.

- *Security challenges and the correctness of data*
  As nodes should work together the importance of the correctness of the data provided by all nodes increases as a single node could potentially influence the results of all other nodes. The correctness can be incurred by for example broken or uncalibrated sensors or an attacker inserting/altering data. The systems is also more vulnerable to other types of attack due the the trust required for a normal operation.

- *Trust in the system*
  Trust in the system is important for both the participants of the collection and the parties that use the collected data. When the trust is low people are less inclined to participate and the parties that use the data are less inclined to use the data as it is less useful. Furthermore the trust in nodes is important in order to let nodes work together to achieve the common goal.

- *Privacy and anonymity of users*
  Lastly privacy might pose a problem in opportunistic networks as nodes are exposed to other nodes, thereby making it possible to trace separate nodes. In using smartphones as nodes it allows tracing of specific users as users won't change smartphones often.

- *Scalability*
  In opportunistic sensing it is important that the solution scales as more or less nodes/people participate over over the lifetime of the system.

- *Semantics*
  In opportunistic sensing there is a lot of data available but it is not clear what it represents. It is important to make sense of the data in order to use it properly.

In this thesis the emphasis is on the scalability, the security challenges and correctness of data in combination with the trust in the system. Privacy aspects and the semantics of different kinds of data is deemed less important for this thesis.

5 Android

The development of Android started in October 2003 when the company Android, Inc. was founded. Initially the intentions for Android were to create an advanced operating system for digital cameras, however the market was to small and the focus instead became the mobile telephone market. The company (and thus Android) was bought by Google in 2005. In 2007 the Open Handset Alliance was formed of several hardware, software and mobile telecommunication companies with the intention to advance open standards for mobile devices [8], and during the founding Android was officially unveiled. The first commercial available mobile phone with Android was the HTC Dream which was released in September 2008, which was introduced with Android version 1.0. Applications for Android are typically Java-based but native code can also be executed.

5.1 Fragmentation

Over the years multiple Android distributions where released, with increasing hardware requirements for newer versions. This caused the Android fragmentation as older phones could not be updated with a newer Android version. Table 1 shows the current Android fragmentation as provided by the developer site of Android. The table only shows Android versions which have at least a distribution of 0.1%.

API Version	Name	Percentage
2.2	Froyo	1.3%
2.3.X	Gingerbread	21.2%
3.2	Honeycomb	0.1%
4.0.X	Ice Cream Sandwich	16.9%
4.1	Jelly Bean	35.9%
4.2	Jelly Bean	15.4%
4.3	Jelly Bean	7.8%
4.4	KitKat	1.4%

Table 1: Android Fragmentation [2] Date: 8-1-2014

These subsequent versions of Android all improved the overall functionality and usability of the operating system, but not all of them are relevant to smartphone mesh networking. The most important updates (in terms of communication methods) are:

- 2.0 - Added support for Bluetooth
- 2.3 - Added support for NFC
- 4.0 - Added support for WiFi-Direct
- 4.1 - Added service discovery to WiFi-Direct
- 4.2.2 - Provided enhancements and increased stability to WiFi-Direct
- 4.3 - Added support for Bluetooth Low Energy

5.2 Current Version

The current version of Android is named KitKat. The first version(4.4.0) of KitKat was released in 31 October 2013, and the current version of KitKat (4.4.3) was released on 2 June 2014. Android 4.4 provided major improvements on memory usage, battery usage and overall changes to improve performance.

5.3 Nature of Android

Android is a open source model, the source code of the operating system is published as soon as a new version is published [1] and everyone can develop & publish applications (publishing to the Android Play store cost money through). Applications can be installed in the Play Store (the Android marketplace for applications), but they can also be installed directly by users however this functionality has to be enabled manually. Android also supports tablets and there is even a gaming console that uses Android as operating system (Ouya).

6 Sensor nodes

In order to collect information on the environment sensors are required. These sensors can be internal (part of a smartphone), external as an attachment (to for example a smartphone) or externally connected to a sensor node. This section focuses on the sensor nodes and the challenges to overcome to use them in the project.

Chapter 7 lists the communication methods available for smartphones, and the most suitable technique for the communication between smartphones and sensor nodes is using Bluetooth Low Energy. Therefore it is decided that it is mandatory that sensor nodes in the project have Bluetooth Low Energy capabilities.

Some market-ready sensor nodes were identified that can be used in the project. These devices are (but not limited to): The LightBlue Bean [5], NODE [4], the CC2531 SensorTag [7] or any other sensor node that contains a Bluetooth Low Energy module.

6.1 Challenges

There are several challenges in using sensor nodes in the project. Most of these challenges are caused by the fact that sensor nodes have limited resources available and it is not desired to replace (the batteries of) the sensor nodes when they are deployed.

Sensor nodes are typically cheap resource constrained devices with limited or no capability to recover lost energy. It is not desirable to replace sensor nodes, thus the energy should be used sparingly to maximize the lifetime of nodes. A way to reduce the effects of the limited power supply is energy harvesting, which is the process of deriving energy from external sources such as solar panels. It should be noted that energy harvesting is not always available for sensor nodes due to the added price or environment factors.

A major part of the energy used in sensor nodes is related to the communication, in either transmitting receiving or idle listening for data. The impact of the communication is depended on the the network traffic, the distance between sender and receiver, the expected interference, the communication technique used and the duty cycle used for the nodes. Other challenges are related to the processing of data as sensor nodes only have limited processing capabilities, and processing consumes extra energy thereby reducing the lifetime of the sensor nodes.

Furthermore there is a trade off in security in sensor nodes as extra security often requires additional processing, however when the security is not sufficient then the trust in the data provided by the nodes is limited, reducing the effectiveness of the network for the purpose of data gathering. There are also some challenges related to the routing of information in sensor networks (consisting of sensor nodes), which are are not relevant to the project as the communication between sensor nodes is omitted in favor of using smartphones for sharing the data.

Lastly there are challenges with the localization of nodes, which is very important in the project as the sensor data is very location depended. The following subsection further describes the proposed approach to counter this set of challenges.

6.2 Location estimation of sensor nodes

During the aggregation the location of the measurement is important to get accurate aggregation results as aggregation is only useful with overlapping data meaning that all measurements should be in a certain distance of one another. For the worst case scenario it is assumed that the sensor nodes have no hardware capabilities to retrieve their exact location. The nodes in the system the sensor nodes can be categorized as dynamic (moving around) and fixed (static location) nodes. Dynamic nodes are, for example, attached to a bike or car and may change their location whilst static nodes are placed at fixed positions and do not move.

The location estimation of a node can be performed by the node itself, the connected smartphone or a combination of both the smartphone and sensor node.

Location estimation by node
It is assumed that the sensor node does not have a way to retrieve the exact location of itself. The node can either be a fixed- or dynamic - location node. To achieve a basic estimation of the location the node is dependent on external sources. The proposed approach is to let the node store previous communication traces to estimate the current location. Further communications with nodes which also know their location can be used to refine the location estimate of the node. Obviously this location estimation is suited better for fixed-location nodes as previous recordings are mostly meaningless for dynamically moving nodes. A stored trace can be, for example, the pair of "Timestamp, node Identifier, Location of known-node, Location Accuracy, Received Signal Strength Indication(RSSI)" retrieved of a smartphone in combination of the connection information, in the example trace it is assumed that the RSSI can be measured by the sensor node. The timestamp is used in conjunction with the node identifier to identify traces of the same node to optionally improve the estimated location using the changing RSSI. The RSSI value is expected to be higher when the connection is better, which can either be the result of environmental factors or the distance of the nodes.

Location estimation by smartphone
A different approach to localizing the sensor node is the estimation by smartphones. This method is suitable for both dynamic- and fixed-location nodes. The behavior is dependend on the availability of neighbor nodes and the communication ranges, identified into three scenarios:

Scenario 1 *There are no neighbor nodes in the communication range of the smartphone.*
The smartphone has to estimate the location of the node all by itself based on the current location of the smartphone, the communication range, the communication RSSI and optionally the history of the location estimations for the node. The RSSI can be used as an indication of the nearness of the sensor node, assuming that a higher RSSI value indicates a nearer node.

Scenario 2 *There are neighbor nodes in communication range of the smartphone but only the current node is in communication range of the node.*
The smartphone can now make a better estimation of the location then in

scenario 1 as it can estimate the direction of the node (opposite the direction of the neighbors). Using the communication rang and the distance between the smartphone and the other smartphones a minimum distance of the node can be estimated. It should be noted that the neighbor nodes *could be* in the communication range of the sensor node however the connection quality could be bad due to external interferences or the sensor node didn't scan for nearby devices.

Scenario 3 *There are multiple neighbor nodes in communication range of both the smartphone and the node.*
The smartphone can now estimate the location even more accurate as the node is in communication range of multiple smartphones. This means that the node is nearby as all neighbors are in communication range. The nodes can share their connection information to improve the accuracy of the estimation.

This approach relies a lot on the accuracy of the location estimation for the smartphone itself. It also depends greatly on the availability and correctness of other nearby smartphones. Compared to the location estimation by node it requires a lot less computational power by the sensor node as all computations and data storage is performed by smartphones. This approach is also more suitable for dynamically moving nodes as the history of the sensor node is not taken into account (except in scenario 1). The approach is less suitable for fixed-location nodes as the history of expected locations (and history of traces) can be far more accurate when a node stays on one location.

Combined estimation location
As the location estimation by node and smartphone are both optimized for either dynamically moving or static nodes it is desirable to have a solution that works for both types of nodes. For this purpose the following approach is proposed, where it is important to detect whether the node is moving (dynamic) or in a fixed location.

In this thesis it is assumed that a node does not have capabilities to detect its own movement (no gyroscope, GPS, compass or acceleration etc) but can be programmed. If the node cannot be programmed and it does not have location detection capabilities then the location estimation by smartphone should be used. When a node has location detection capabilities they can be used instead of the proposed method in this chapter. The classification of dynamic or static can be achieved by using the history of communication traces to check if the node has moved too much, indicated by having an estimated position that differs more than the communication range plus inaccuracies with a previous estimation, thereby ensuring that a node has moved. On detecting the change the node will be assumed to be a dynamic node as it must have moved or the accuracy of the estimation was really bad.

1. The node will try to estimate its own location (see Location estimation by node).
2. The node sends its estimated location when requested by the smartphone.

3. The smartphone checks if the reported estimated location is within its communication range (validity check) and if the reported estimation accuracy is good enough.

4. When a node is classified as dynamic then the freshness of the location estimation (how long ago was it updated) is also checked to make sure it was recent enough.

5. If the checks fail then the location is estimated by the smartphone, otherwise the reported location of the node is used.

This approach is both suitable for dynamically moving nodes and static nodes as the approach combines both approaches depending on the classification of the node. Parameters in this approach are: the time an estimation is deemed recent, the communication range and the classification strictness (how much can a node move whilst staying classified as static).

7 Inter-Smartphone Communication

An important part of the project is the communication between smartphones in order to realize sharing of sensor data and aggregation results. An easy solution would be to send the information to an external server and retrieving the data by a different client, however an internet connection is not always available rendering the the solution useless as it should work without internet, thus other communication methods should be identified and used.

There are several communication methods available that can be used for Android smartphones to inter-communicate, however the selection of techniques which do not require an infrastructure or additional hardware whilst having sufficient range is limited. To determine how suitable the communication methods are 4 aspects are defined and taken into account:

- *Transmission range*
  What is the transmission range of the technique? For sharing the data a long range is proffered.
- *Availability*
  Is the technology available on all phones, or just newer/older phones? It is desired that the technique is broadly available.
- *Ease of access*
  How hard is it to use the technique, can it be using without interaction of the user, and how well does Android support the technique? It is desirable to have a technique that is easy to use, both for the user as programmers of the system.
- *Capacity*
  How many connections does the technique support, is it one to one or many to many? It is desirable that the capacity is in a many to many relation to simplify sharing.

Table 2 shows the comparison on the suitability of the communication methods to be used in smartphone mesh networks according to the 4 aspects. It appears that Bluetooth and WiFi are the best methods for the communication. Bluetooth has a lower range compared to WiFi, which might prove problematic in low-density areas, however it may be used at the same time a WiFi communication is used. The WiFi solutions have the problem that they cannot connect to normal WiFi networks when being used (except for WiFi-Direct, and even then it is optional). A combination of Bluetooth and WiFi is possible, which combines the positive sides of both solutions at the cost of (possibly) battery power.

Bluetooth Low Energy could also be used for smartphone to smartphone communication however it might interfere with the smartphone to sensor node communication, so it was chosen to not use this technique for the inter-smartphone communication in favor of Cocoon.

7.1 Bluetooth Low Energy

Bluetooth Low Energy, also called Bluetooth smart or Bluetooth LE, is a wireless personal networking technology based on (but not backwards compatible

Name	Transmission range	Availability	Ease of access	Capacity	Overall score	Notes
WiFi infrastructure	++	++	-	++	+5	Cannot use normal WiFi whilst using this method
WiFi Ad-Hoc	++	-*	+*	+	+3	Requires rooted Android device
WiFi-Direct	++	+	-	+**	+3	Exact capacity unknown
Bluetooth	+	++	+	+	+5	Can also be used when connected to WiFi
Bluetooth LE	+/++	+	+	+	+4	Low energy consumption, requires Android 4.3+, varying range per device

Table 2: Technology suitability for current Android smartphones
Only the most suitable methods are shown
* depends on rooted/non-rooted
** Exact capacity unknown [6]

with) Bluetooth. The major improvement over Bluetooth is the reduction in power consumption whilst maintaining a similar or larger communication range. Bluetooth Low Energy is supported on Android version 4.3+ however not all chipsets used by smartphones support Bluetooth Low Energy. The standard was integrated into the standard Bluetooth Core Specification V4.0. Bluetooth Low Energy nodes communicate in a client-server style: The GATT client initiates commands and requests whilst the GATT server services the requests and responds to the GATT Client.

There are several terms in Bluetooth Low Energy which are relevant for this thesis:

1. Attribute protocol (ATT): Communication method which is optimized for small packet sizes used in Bluetooth Low Energy, allows an attribute server to expose a set of attributes with associated values to an attribute client

2. Generic Attribute Profile(GATT): Core of Bluetooth Low Energy, build on top op the Attribute Protocol, and establishes common operations and a framework for the data transported and stored by the Attribute protocol.

3. Client: Device that initiates GATT commands and requests, which accepts responses.

4. Server: Device that receives GATT commands and requests, which returns responses (like sensor data).

5. Profile: Description detailing how a device works in a particular application, a profile may contains multiple services.

6. Service: Collection of data and associated behaviors to accomplish a particular function or feature of a device, it may contain multiple characteristics and optionally other services.

7. Characteristic: Contains properties, value(s) and descriptors which is transferred between client and server

8. Descriptor: Extra attributes of an characteristic (for example the unit of measurement, range properties or represented format)

The theoretical maximum range of Bluetooth Low Energy is set at over 100 meters [32], however most commercial products found have a maximum range of either 100m (WaspMote BTLE module) or 150m (Anaren Integrated Radio) outdoors while requiring line of sight. In a press release Bluetooth SIG states

> "The range of the Bluetooth v4.0 radio may be optimized according to application. The majority of Bluetooth devices on the market today include the basic 30 foot, or 10 meter, range of the Classic Bluetooth radio, but there is no limit imposed by the Specification. With Bluetooth v4.0, manufacturers may choose to optimize range to 200 feet and beyond, particularly for in-home sensor applications where longer range is a necessity" [9]

7.2 Cocoon

Community-oriented Context-aware Opportunistic Networking (Cocoon) [35] is a technique that provides a WiFi infrastructure by dynamically turning a smartphone into a WiFi hotspot or a WiFi client (connected to a Cocoon network) depending on the number of available Cocoon networks and preferences. It is a new protocol developed by the Pervasive Systems group on the University of Twente. All references and information on the protocol are based on the current state (Cocoon is not yet published). Cocoon currently has 5 modes of which 3 can be used to create smartphone networks (global does not create a network and smart switching is not yet available):

- Global - Cocoon is disabled, the smartphone can connect to normal networks
- Client - The smartphone will always be a client in a network, it will not switch to hotspot mode. When not connected to a Cocoon network the smartphone will try to connect to one.
- Hotspot - The smartphone will always be a hotspot in a network and broadcast the Cocoon network. Other nodes can join this network.
- Switching - The smartphone toggles between client and hotspot modes after a specified (adjustable) time interval.
- Smart switching (Future work) - The smartphone will choose a mode depending on past experiences. If there are no networks available it will become a hotspot. If there are some networks available it will connect

to them. When acting as a hotspot in this mode the number of clients reduces the chance that it switches to client mode.

Cocoon networks are identified by the name which equals "Cocoon-" followed by a number from 0 up to 10. In the current version of Cocoon the smartphone will always connect to the network with the highest number available when finding available Cocoon networks.

A downside to using Cocoon is that the smartphone cannot use regular WiFi at the same time as Cocoon (except when in global mode).This reduces the chance that a smartphone is connected to the internet whilst connected to Cocoon as the hotspot has to share its mobile-subscription internet. Cocoon is currently only available for Android version 3.2+.

8 Architecture

As part of the research a prototype should be designed and implemented to demonstrate the feasibility of using smartphones in environmental monitoring in the context of urban-sensing. This chapter describes the design for the prototype, which was also implemented. Chapter 9 provides the implementation specific details of the prototype including screenshots of the application.

8.1 Layering

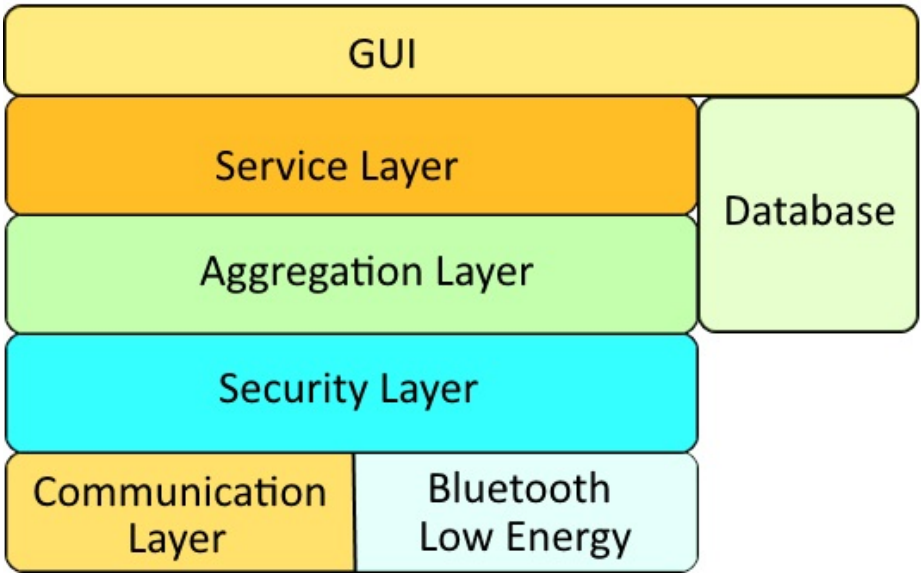


Figure 2: Architecture Layering

The functionality of the architecture is divided into multiple layers which are described in this chapter and is visualized in figure 2.
The first layer is called the"Aggregation layer", which is responsible for aggregating the received sensor data and storing the data.
The second layer is called the "Security layer", which is responsible for all security related issues like authentication of
nodes, authenticity of messages and prevention against replay attacks.
The third layer is called the "Communication layer", which is responsible for the network and neighbor awareness and transport & integrity of the messages to send.
The fourth layer is the "Service" layer which is responsible for the collection of data like the node's location, controlling Bluetooth Low Energy and managing the settings in the system.
The fifth layer is the database, which provides database access for storage of sensor data and querying of stored data.
The last layer is the GUI, which is the graphical user interface the user sees and controls.

Once a message is received in the communication layer the first processing is performed. When the packet is valid (and the application should receive it) then the parsed packet is passed to the security layer which does more processing and finally (if the message is still correct) is passed to the aggregation layer. The aggregation layer will act depending on the type of message. When it is raw sensor data (sharing mode) then it is added to the list of recent data and optionally (when the aggregation is not active) starts the aggregation on the node. When it is an aggregated result the node will check if it needs the data or not. When it is required the data is stored otherwise it is discarded. Lastly the message type can be a certificate (as part of the security layer). In this case the message is discarded as it should be filtered out by the security layer. The prototype only allows one instance of each type of layer at a time.

8.1.1 Communication Layer

The communication layer is responsible for the inter-smartphone communication. This layer is heavily depended on lower-layer communication APIs of Android. For the prototype 3 types of the communication layers are implemented which can be used for inter-smartphone communication: Loop-back, Cocoon and Bluetooth.

Cocoon
In the prototype Cocoon can be used for Smartphone to Smartphone communication. It is possible to select which mode to use of Cocoon (Hotspot, Client, Switching).

Bluetooth
There is also an Bluetooth - communication layer implementation. The Bluetooth implementation has 2 modes of operation: Discovery and Paired mode. In discovery mode the node will constantly try to find new nodes in range and will transmit data to all nodes in range. In paired mode the node will not discover new nodes but will only try to send data to paired nodes (when they are in range).

Loop back
In the prototype a loop back adapter is available for easy testing. The loop back adapter links the send and receiver methods of the communication layer interface thus routing outgoing messages back to the receiving methods. The implementation does not provide network awareness as it doesn't connect with any other node.

8.1.2 Security Layer

A publicly used system can be subject to attacks, for example replay attacks or a masquerading attack can be performed. To reduce the vulnerability to attacks the Security Layer is introduced. The goal of the Security Layer is to provide message authenticity and prevent against a predefined set of attacks.

Certificate based Security Layer
In the prototype there is but one security layer. The implemented security layer provides message authenticity by using lightweight- certificates signed by a trusted authority. The nodes are responsible for creating their certificate and sending it to the trusted authority which signs the certificate. The prototype uses the Digital Signing Algorithm (DSA) to sign and verify the messages and certificates. The lightweight certificate in the prototype contains:

- The identifier (name) of the node (provided by the node)
- The expiration date of the certificate
- The authority which signed the certificate
- The public key of the node specified (provided by the node)
- The signature of the certificate (provided by the trusted authority)

The lightweight certificate was chosen (as opposed to for example an X.509 certificate) to reduce the size required. It would be reasonable easy to migrate to a normal X.509 certificate when the need arises, however this is not yet needed as a lot of parameters in the normal certificate are redundant when used in the application such as which algorithm is used for the public key and certificate signing. The assumption is here that a certificate is valid from the moment it is signed until the expiration date of the certificate.
When nodes receive a certificate during normal operation (without internet connection) then the certificate can be verified using the known public key of the authority to check the authenticity of the certificate. When it is authentic the public key contained in the certificate can be used to check the authenticity of messages received by the corresponding node of the certificate.
The implemented Security Layer prevents against masquerading and data injection attacks as the source of the messages is authenticated using the certificate. Prevention against replay attacks is achieved by using a timestamp in the messages. A Sybil attack is still possible but is a lot harder to execute as all (attacking) nodes have to be registered at the TA. When the data is corrupted the authenticity check of the message will fail, however the data cannot be correctly recovered by the node.

Benefits of the Certificate based Security Layer
The security of the system is improved by enforcing authenticity of messages. Furthermore known attackers can be blacklisted, and new nodes can be denied access to the network as the TA has to provide a valid certificate to let the node participate in the network. The implemented security layer prevents against masquerading attacks and data injection. Sybil attacks are harder to execute as all attacking nodes have to be registered at the TA, and in combination with the blacklisting some attacking nodes can be filtered out. Lastly the layer ensures data integrity as the authenticity check will fail when the message is modified. The layer does **NOT** provide message confidentiality as all nodes should be able to read the messages of this node.

8.2 Location estimation of sensor node

An important part of the data accuracy is the estimation of the sensor location. In the ideal case the node knowns its exact location, however it is expected that sensor nodes may not have the capabilities to determine their location, thus location estimation is required. In the prototype the basic principles of chapters 6.2 (location estimation) and 8.6 (sensor to smartphone) are used in combination with some extra filtering described here.

Step 1 Remove all traces that are too inaccurate (configurable parameter) or too far away from the estimating node (distance is configurable).

Step 2 Add X to all node's accuracy value, where X is $0.5 * MAX(distance_{i-j} - accuracy_i * 2 - accuracy_j * 2)$ when X is positive, otherwise the scaling is not required. In this formula i and j are any 2 of the trace-locations.

Step 3 Check if a circle exists completely in a different circle: If yes remove the encapsulating circle

Step 4 Calculate all circle crossings (1 or 2 per circle), when there is only 1 crossing detected then the target location *should* be at that position if there where no measurement errors.

Step 5 Remove all crossings that do not overlap with all circles.

Step 6 The estimated location is the average position of the minimum and maximum of all accepted circle crossings, where the minimum is the minimum latitude and longitude and the maximum location has the maximum latitude and longitude.

Step 7 The estimated accuracy circle radius (100%) is the maximum distance to all accepted circle crossings.

8.3 Aggregation Layer

To process the collected data and perform the aggregation related tasks in the application some processing is required which is performed by the so called aggregation layer. This layer is responsible for handling the received messages from the security layer, aggregating the data, storing the data and optionally sharing the data.

Currently one aggregation layer is implemented called the 'Basic Aggregator'. This layer implementation provides the basic aggregation operations and uses the aggregation cycle as described in chapter 8.3.1. When a packet is received the layer will try to store the received data into the measurements database followed by optionally aggregating and sharing the data depending on the current mode of the aggregation cycle.

The aggregation in the prototype currently only supports two types of data: locations and numeric values. The aggregation result of the numeric values results in a composite variable containing the average, minimum and maximum values of the list to aggregate. The aggregation result of the locations is more complex, supporting two possible aggregation functions: the location-estimation algorithm as described 8.2 and a function that tries to provide information on the location accuracy by reporting the average location, the sample deviation

of the location and the root mean square of the location. For the freshness of the data for the aggregation the freshness function as defined in section 8.8.4 is used.

8.3.1 Aggregation Cycle

To reduce the data set size the aggregation cycle is introduced. The aggregation cycle starts as sensor data is available, initiating the sharing of the sensor data with neighbors for a specified duration. After the sharing of sensor data the aggregation is performed, followed by sharing the aggregation results for a specified duration. After the aggregation result sharing period the node will wait a specified amount of time before being able to start the aggregation cycle again.

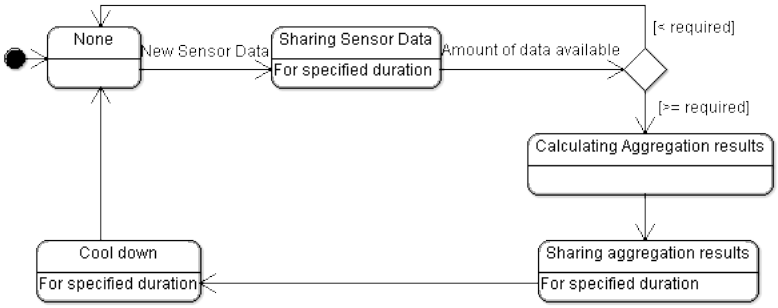


Figure 3: The states of the aggregation cycle

The parameters used in the aggregation cycle (as shown in Figure 3) are:

- The minimum number of elements required in the aggregation,
- The duration of the 'sharing sensor data' state,
- The duration of the 'sharing aggregation results' state,
- The duration of the 'cool down' state,
- The number of broadcasts during the 'aggregation sharing' state,
- The number of broadcasts during the 'sensor data sharing' state.

The effects of these parameters are very depended on the environment but basically the most important parameter is the minimum number of elements required, fewer items required causes more aggregations to be performed. The number of broadcasts parameters influence the number of messages send, a higher value ensures more messages will be send to neighboring nodes. The cool down duration parameter is important to reduce the number of aggregations but at the same time it may cause problems as neighboring nodes may have

conflicting states of the aggregation cycle thereby not sharing data at all anymore. The effects of some of these parameters were further validated in section 10.4.

8.4 Database storage

To store raw and aggregated sensor data a local database is used. The database provides a simple mechanism to efficiently store and retrieve data. In the prototype a SQLite database is used as Android has built-in support for this database. The data format used to store complex structures of data is in database fields is JSON. The default storage format of sensor data is: {*"type":<type>,<element data>*}. Where type is the data type and element data is the type-specific payload which is also JSON-encoded.
A location would be stored as {*"type":1,"latitude":6.856384025, "longitude":52.239406175,"altitude":0.0,"accuracy":55.5,"time":1396259407847,"provider":"gps"*}.
A composite variable of one location would be {*"type":3,"data":*[{*"type":1,"latitude":6.856384025, "longitude":52.239406175,"altitude":0.0,"accuracy":55.5,"time":1396259407847,"provider":"gps"*}]}.

8.5 External server

A laptop with a Apache web-server installation with PHP5 was used as an external server. This server was responsible for receiving the aggregated data and acting as the Trusted Authority. Only a simple version of an external server was made which stores the uploaded data in a MySQL database.

The external server may sign certificates uploaded by nodes so that they can operate without internet connection after receiving the signed certificate. The external server verifies the node's certificate to check if a node is authorized to upload the data. A simple website has been made which presents the user of the external server with a visual representation of the collected data.

8.6 Sensor node to Smartphone communication

For the connection between the sensor node and the smartphones Bluetooth Low Energy was used (Chapter 7.1).
The sensor node runs the Bluetooth Low Energy GATT - server, which means it receives and processes data requests.
The smartphone runs the Bluetooth Low Energy GATT - client, which initiates the connection, requests data and receives it.
The smartphone scans periodically for Bluetooth Low Energy nodes in range. When it detects a node nearby it tries to connect to the node and retrieve the recent sensor data and the location information. The location information can either be an exact location (if the node has capabilities to determine its location) or a series of recorded location broadcasts of previously connected smartphones. The smartphone can then try to estimate the location of the sensor node based on its own location and the reported locations (using combined location estimation as described in chapter 6.2).

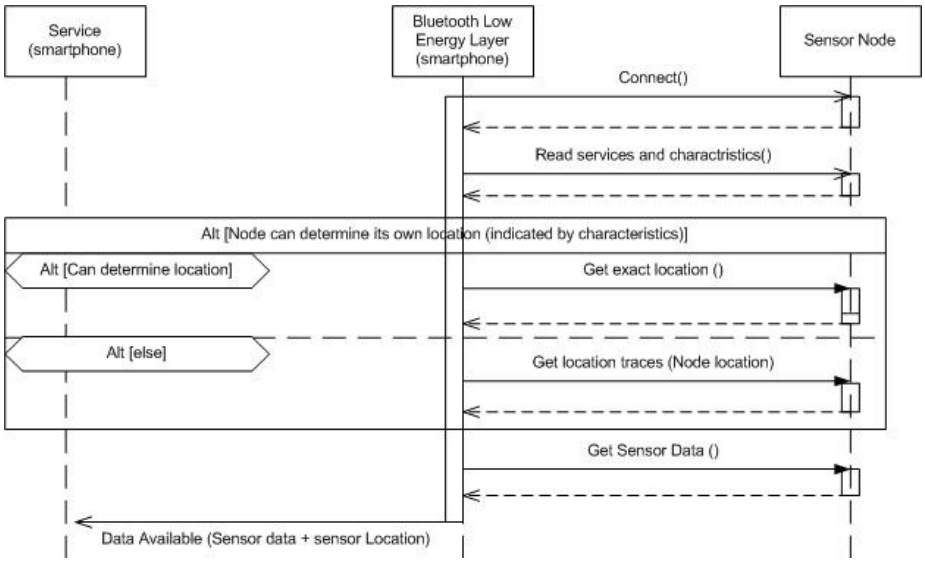


Figure 4: Sequence diagram - Sensor node data retrieval

8.7 Smartphone to External server

In order to use the collected informations on smartphones by other parties than the user it is required that the information is send to a server. The information that is uploaded can either be raw sensor data or aggregated data.

Uploading data, both aggregated and raw data, has some challenges which should be kept in mind:

1. Size: Size of the table to maintain in a node which should be stored for upload. When the size of the data to upload becomes to large then users may complain as smartphones only have a limited data storage which is filled by the application and cannot be used at the same time by other applications. A too small storage size however causes loss of data as it can no longer be stored.

2. Detection of falsified data: How easy is it for the external server to detect falsified data using the history of all uploaded sensor data. Falsified data may have huge implications for the trust in the system as false data makes the system untrustworthy and thus useless for third parties.

3. Privacy of the data: How much privacy-related content does the information contain? As the results are uploaded to an external server some privacy issues may occur as the information could be used to track the movement of participants. The aggregation will omit some of the privacy-related data like the exact sensors it came in contact with (and on what time), however some traceability will remain.

4. Data accuracy: How exact is the data uploaded? It is important that the data is as accurate as possible as accurate data represents the environment far better than inaccurate data. When aggregation is performed some individual accuracy will be removed and a more global accuracy will be

created. To make sure that only useful data is present on the server it could filter uploaded data based on the accuracy.

Table 3 summarizes the effects of the type of data that is uploaded on the challenges. Uploading the position is required to indicate to which area the results belong. Optionally the accuracy of the position can be reduced increasing the privacy and reducing the data accuracy and reducing the detection of falsified data.

Characteristic	Aggregation and position	Sensor data and Position
Size	++*	–*
Detection of falsified data	–	++
Privacy of Data	+	-
Data accuracy	-*	+*

Table 3: Trade-offs of data aggregation in the upload to an external server
* Depends on the aggregation ratio and data set size

8.7.1 Attacks on the upload

As the system would be publicly available some analysis on the possible attacks on the webserver (related to uploading the data) is required. The attacks can be categorized into:

- Sybil attack: Pretend to be more than one node
- Masquerading: Pretend to be a different node
- Replay attacks: Resend old messages to the server
- Selective forwarding: Only send selected or no messages instead of all messages
- Node compromise: Attacker gains full control of a node, thus can upload anything
- Data corruption/injection: The (sensor) data gets corrupted (during transmission or otherwise).
- Denial of service: The attacker performs an attack on the webserver with the intent of disabling the server, thus making uploading of data impossible.

To prevent some attacks and reduce the effects of attacks some steps can be taken.

1. The first option is to limit the access to the upload by using whitelisting. This ensures that only registered users can attack, limiting sybil and masquerading attacks.
2. The second option is to perform pre - and/or post processing on the uploaded data. By comparing the uploaded data with data of nearby nodes or previous recordings a 'normal' state can be determined. If the uploaded

data differs a lot from the normal state then an event occurred or the data is corrupted. This option reduces the effects of a node compromise and data corruption.

3. The third option is to use signatures in combination with the data. When a node signs the data (including a timestamp) with a private key the data can be checked using a public key. As long as the private key remains a secret the messages of a node cannot be faked or altered (data alteration is detected). This option eliminates masquerading data corruption and replay attacks (due to the timestamp). It does not prevent a node compromise as the attacker will know the node's private key after the compromise. Furthermore it will not prevent sybil and selective forwarding attacks as an attacker may have compromised multiple nodes.

4. The fourth option is to use a witness scheme to provide proofs of the data. The upside of this option is that it is harder to perform a sybil attack as more nodes have to be compromised, data corruption can be partially detected as the data can be checked with the proofs. The major downside to this option is that multiple nodes have to participate otherwise no proofs can be generated. An attacker can still 'fake' proofs unless encryption or signatures of devices are used.

5. The fifth option is to use a blacklist, which prevents against known attackers. This option is not very useful as nodes do not have to update their blacklist often and when attackers use masquerading attacks the blacklist will be avoided.

The prototype uses a combination of whitelisting and signatures to reduce the attack risk as sybil attacks, masquerading, replay attacks and data corruption are prevented. Furthermore it prevents a bit against selective forwarding attacks as all nodes that took part in the aggregation may upload the data. It is not possible to prevent against denial of service attacks. As an effect of the signatures the nodes have to initially let the server sign their certificate so that other nodes can check the node's authenticity.

8.8 Aggregation

An important part of the application is the aggregation of sensor data. The most important reasons for aggregation is combining sensor data of correlating nodes to improve their data as sensors occasionally report inaccurate or incorrect data which can be filtered out by comparing it with correlating data of neighbor nodes/sensors. This results in an increase of global accuracy at the cost of individual sensor accuracy. Another important reason is the reduction in data size as redundant data can be removed after the aggregation. There are also some side effects of the aggregation of which some are positive and others are negative:
Upsides of aggregation:

- *Increased global accuracy and average*
  Combining the sensor data provides a better representation of an area compared to the findings of a single node.

- *Reduce data set size*
  By combining the sensor data redundancy can be removed, thereby reducing the data set size.

- *Increase anonymity*
  As individual information of nodes is removed by aggregation some anonymity is introduced as the sources of the aggregated data cannot be deduced.

- *Performance of using the data*
  Operations that should be performed on all data will be faster on a smaller set size, thus aggregation will speed up these operations as the data set size is reduced.

There are also some downsides of the aggregation:

- *Decreased local accuracy*
  By using an average the individual measurements of a node are lost meaning that the local accuracy is deleted and replaced by a more global value.

- *Not possible to track or verify specific measurements*
  As individual measurements are removed it is no longer to track or inspect specific measurements of nodes.

- *Reduced data set*
  In some cases it is desirable to have a larger data set with raw data. This is no longer possible because of aggregation.

- *One time performance penalty to aggregate the data*
  Aggregating the data required processing of the data. Although it is only once per aggregation it takes a while for each aggregation, depending on the amount of data to aggregate.

- *Hard to trace the sources of the aggregated data*
  The aggregation removes individual traces of nodes, making it impossible to remove traces of individual 'corrupt' nodes.

8.8.1 Problem analysis

There are several factors which are relevant to the design of the aggregation. First there are the environmental and contextual parameters.

- *Mobility*
  An important factor is the mobility of the users and sensor nodes, which is in this scenario classified as 'medium' as the speeds vary from standing on a static position up to public transportation. The expected speeds are 0 km/h for a static position, 5 km/h for walking, 20 km/h for cycling and 25km/h for bus average speed. The mobility might pose a problem as nodes can move a long distance in a short time period as depicted in table 4. The aggregation should ensure that only relevant sensor data is aggregated of a certain area instead of only relying on the time between measurements.

Speed km/h	1 second (m)	1 minute (m)	5 minutes (m)	20 minutes (m)
5	1,388	83,28	316,4	1665,6
20	5,55	333	1665	6660
25	6,944	416,66	2083,33	8333,33

Table 4: Maximum distances of movement speeds of nodes

- *Node Density and communication range*
  The amount of nodes in an area is important as more nodes can take participate in the aggregation. In this scenario the density of the sensor nodes is estimated to be varying between no nodes and several nodes depending on the interrest in a region. The smartphone density will vary a lot however it is assumed to be medium density (during daytime hours) as the communication methods between smartphones have a long range (up to 100m) and in cities there are a lot of people who can participate.

Secondly there are the data and message specific parameters which are relevant:

- *Message confidentiality*
  It is assumed that there is no message confidentiality as all other nodes can read the shared data on the environment.

- *Data integrity*
  Message and data integrity is very important as data should be trustworthy, otherwise the project is meaningless as all data would be potentially fake.

- *Message authenticity*
  Message authenticity is deemed important as it is required to prevent data injections and ensure integrity of all data in the system.

- *Message and sensor data freshness*
  In aggregating sensor data it is important that the sensor data correlates. In this thesis the correlation is determined by 3 aspects: The type of sensor data, the location of the data and the 'age' of the sensor data. More details on the freshness of sensor data can be found in chapter 8.8.4.

- *Data rate*
  It is expected that the data rate is low on average it will have bursts of information when an aggregation is initiated (sharing of the data) and when the aggregation is completed (sharing results).

- *Message size*
  It is expected that sensors provide small packets with data resulting in the aggregation of small data packets.

Lastly there are some attacks that can be performed on the aggregation which should be kept in mind:

- *Replay Attacks*
  An obvious attack that can be performed on the aggregation is the replay attack. In this attack old data is shared so it will be used in the

aggregation. The old data should not be used as the data is no longer respresenting the current situation. A simple solution would be to enforce using timestamps. These attacks are related to the freshness parameter.

- *Data injection and corruption*
  The second and third attack are the data injection and corruption. These attacks either alter existing information or inject new false information, resulting in an incorrect aggregation. These attacks relate to the Data integrity and authenticity parameters.

- *Node compromise*
  An attacker can compromise a node, totally controlling its behavior. The attacker can use the compromised node to perform other attacks or use it for monitoring of the network. The message authenticity should minimize the damage as only one node is compromised which cannot impersonate other nodes.

- *Masquerading*
  The last attack is the masquerading attack where a compromised node tries to impersonate a different node. This should be prevented by the message authenticity.

A network of nodes can be classified as either a flat or hierarchical network. In a flat network all nodes behave the same way, there can (optionally) be no specific root or controlling nodes. In hierarchical networks nodes can have different tasks like sensing, relaying or processing but they may also perform all tasks. Often in hierarchical networks there is one root node which performs the processing and several relaying nodes. In the current setting nodes perform all tasks (measuring, relaying information aggregation and optionally upload to the external server). The nodes operate in a dynamic environment with a changing number of neighbors as nodes move around. During operation the nodes are not connected to any central sink in the system (during the upload to the external server the node requires an internet connection, and might not take part in the normal operation). Furthermore the operation of the system should not be hindered by a lack of internet connection of several nodes and/or a specified amount of time.

8.8.2 Existing solutions

Aggregation of sensor data is not a new topic as most sensor networks need aggregation to reduce the network traffic. This chapter lists some relevant or promising aggregation solutions. A large part of this chapter is based on the work of the authors of [30] who published a survey which lists some data aggregation techniques for both hierarchical and flat networks. The paper separates the aggregation techniques into 3 major categories: Network-architecture based, Network-flow based, and Quality of service aware methods.

Network architecture-based

Flat-Push Diffusion
In the push diffusion scheme the sources flood the data when they detect an event while the sinks in the network subscribe to the sources to receive the data.

A well known family of protocols that use push diffusion scheme are the 'Sensor protocol for information via negotiation (SPIN)' protocols [27]. SPIN uses two techniques to improve on classic flooding: negotiation and resource-adaption. In the negotiation the descriptors of the data are referred as meta-data. The resource adaption is achieved by querying the resource manager so that it can optionally favor different activities like forwarding data instead of measuring data. There are 3 stages and types of messages in SPIN:

- ADV - New data advertisement which is send when new data to share is available, the ADV contains the meta-data.
- REQ - Request for data which is send when a node wishes to receive data, it contains the meta-data of which the sensor data is requested.
- DATA - Data message which contains the sensor data with a meta-data header.

When the data size $\leq$ meta-data size then the ADV and REQ phases are skipped and the data itself is send to save bandwidth.

An attempt to add security to SPIN is Secure-SPIN proposed in [37]. Secure-SPIN is focused on cluster-based networks with a sink, in which the cluster-head knows the private key of the nodes in its cluster. It uses session keys distributed by the sink to guarantee freshness of data and prevent replay attacks.

Overall SPIN appears to be usable for the aggregation as different smart-phones may or may not be interrested in different types of sensor data. Furthermore the Secure-SPIN methods aren't usable as fixed- clusters are not used.

Flat-Two Phase Pull Diffusion: (directed diffusion)

Two phase pull diffusion as described in [24] is based on broadcasting of an 'interest' message by a sink which is relegated throughout the network to determine the direction the data has to be send and the transmission of the sensor data. This method is suitable for a scenario with few sinks and a lot of sources, but is not suitable for continuous data delivery to the sink. This method may not be suitable in low-density networks and might not be effective when all nodes request the same data type.

Flat-One Phase Pull Diffusion

One phase pull diffusion is an improvement on two-phase pull diffusion to remove some of the overhead caused when there are many sources and sinks. This is done by only-transmitting to the lowest-latency node in the path from the data source. This method outperforms two-phase diffusion on a high event rate but two-phase diffusion outperforms one-phase diffusion when the sink has a high interest rate. This method may also not be suitable in low-density networks and might not be effective when all nodes request the same data type.

Hierarchical-Cluster-based

In Hierarchical cluster based networks the techniques that are available use a local aggregating node or cluster-head for aggregation and communication with the sinks (either direct or via multi-hop).

Low Energy Adaptive Clustering Hierarchy (LEECH) :

In LEECH nodes organize themselves in clusters and cluster heads fuse the data periodically and send it to the sink. LEECH is suited for constant monitoring and periodic data reporting. There are two main phases in LEECH: the setup

phase, which is responsible for setup of the clusters and the steady state in which the cluster heads perform the data fusion and sending the data to the sink.

Hybrid Energy Efficient Distributed Clustering Approach (HEED) :
HEED operates similar to LEECH but HEED focuses on forming efficient clusters for maximizing network lifetime. This is achieved by cluster head selection based on residual energy and a proximity (to neighbors) factor.

Clustered Diffusion with dynamic data aggregation (CLUDDA) :
CLUDDA combines directed diffusion with clustering. In this method only the cluster heads take part in the directed diffusion instead of all nodes. In CLUDDA the aggregation points are dynamic, as all cluster heads/gateway nodes may perform the aggregation.

LEECH, HEED, CLUDDA all use clusters which may or may not be effective in a testbed as all smartphones operate independently and there is no fixed cluster head thereby making CLUDDA the most suitable from the hierarchical-cluster based techniques.

Hierarchical-Chain-based
In hierarchical chain-based networks the nodes transmit only to the closest neighbor in the direction of the sink. This approach limits extra power required when cluster heads are farther away in the network.

An hierarchical chain-based method is Power efficient data gathering protocol for sensor information systems (PEGASIS). In PEGASIS nodes are aligned in a linear chain for data aggregation. The farthest node from the sink will initiate the data gathering, as each node towards the sink will receive receive data from nodes farther in the chain, which it then aggregates and sends along in the chain towards the sink.

Hierarchical-Tree based data aggregation
It is also possible to order a network into a tree-based fashion. The sink will be the root of the tree, and contains up to 2 children. The children can again have up to two children. The lowest level (the farthest from the sink) are the leaves, which don't have children. In tree-based networks a node is responsible for the aggregation of the data from the children of the node.

Hierarchical-Grid based data aggregation
Grid based networks are very similar to cluster based networks. In grid based networks there is a node which aggregates and sends data for the nodes of the grid. The node with the strongest signal will send the aggregated data towards the sink. The major difference with cluster based networks is that in a grid based network the nodes do not communicate with each other as opposed to a cluster-based network.

Network flow-based
It is also possible to do aggregation based on the network flow instead of the locations of the nodes.

Lifetime-maximization based
This class of protocols tries to maximize the lifetime of the network based on the (current and expected) network flow, often based on heuristics. This class is not very promising for the project as the network flow is varying a lot depending on the data available and the number of smartphones which are near.

Quality of service aware data aggregation
The QOS-based methods are based on the on network flow methods, however now the performance of the aggregation and forwarding is involved. There are 2 major approaches: The first is to maximize the amount of information collected at the sink, the second is to focus on congestion control and end to end reliability. These methods are just like network flow-based methods not very suitable for the project due to the unreliable network flow.

Secure data aggregation
To prevent against attacks on the aggregation some security alterations can be made to the aggregation schemes. A large problem is that once falsified data is aggregated it is hard to distinguish it from normal data. There are several proposed solutions, but many are specialized for hierarchical/tree based networks instead of flat networks or try to ensure message confidentiality.

The authors of [11] performed a survey on Secure data aggregation models which have a querying node which should receive the correct information. Most of these protocols are designed for a tree-like environment.

The authors of [29] propose a framework for secure data aggregation in large sensor networks using random sampling with proofs (commitment). It tries to prevent false data insertion and does not assume a node to be honest.

Witness based approach for data fusion assurance
The authors of [20] proposed using proofs of witnesses to verify the aggregation was correct. It is based on sharing the data to aggregate with neighboring nodes which all calculate the aggregation. Then the aggregating nodes broadcast the message authentication code (MAC) of the aggregated result which in turn is stored by the node which may want to upload the data. On uploading the aggregated data the MACs are used as proof of the aggregation-correctness. For this principle to work it is important that all aggregating nodes use the same measurements and parameters. It is also required that there are at least multiple other nodes to participate in the aggregation, otherwise the proof cannot be collected. This approach may not be usable as nodes may have received only a subset of sensor data which differs per node.

Relaxed authenticity for data aggregation in WSN : ESAWN
The authors of [15] describe their approach for relaxed authenticity for data aggregation in wireless sensor networks: ESAWN. ESAWN is a tree based aggregation of which the bottom n-k levels of the tree are aggregated and the validation of correct aggregations is performed by each node's parent).
This approach is not suitable for the design as it requires nodes to be represented in clusters of trees. Another downside is a simple attack where the uploader is only parent with all other nodes as children, in this case the attacker can fake all information (large scale injection/data corruption).

Brooks-Lyengar algorithm
An algorithm to exchange data with accuracy is the Brooks-Lyengar algorithm [16]. The Brooks-Lyengar algorithm is a hybrid algorithm that combines data fusion with Byzantine agreement to filter out-of-range values and average the accepted values. The rough steps of the algorithm are as follows:

1. First the node shares the data to aggregate with its neighbors
2. The second step is to filter out invalid (out of range) information, which results in the lower and upper bound and accuracy of the accepted values.
3. The final step is to calculate the weighted average of the accepted values, where the weights are the number of sensors whose readings intersect with each of the accepted values.

An attacker can falsify the aggregated result, but it won't hurt the normal operation of the system.

8.8.3 The SDSN-Aggregation

The SDSN aggregation method uses a combination of methods "Brooks-Lyengar algorithm" and "Witness based approach for data fusion assurance" to achieve the aggregation. The aggregation method should work for multiple data types and for different densities of the network. The SDSN-aggregation is event-based, it is triggered on an aggregation request by another node or when new data is available. All data types have specific parameters for the freshness function which indicates if the sensor data is still fresh enough for aggregation or not. To use this implementation it is important to known the location of the sensor data and the time of the sensor data.

The aggregation is triggered when new data is available for aggregation and no recent aggregation was performed (viewpoint of the initiated node):
Where the variable X is the setting indicating the number of required proofs of the aggregation, and the variable Y is the data sharing timeout indicating how long the node should wait for other nodes data sharing.

1. The node checks if there are at least X neighbors available which can perform an aggregation. When this is not the case then a node will wait until there are enough nodes available or until the data is not fresh anymore, in the last case the aggregation is aborted but the raw data is stored.
2. The initiator node broadcasts its available fresh data to its neighbors with an aggregation request and waits for period Y. During this period other nodes can broadcast their local (fresh) data) to additionally be used for aggregation.
3. The nodes filter out incorrect and non-fresh data and start the aggregation (using the Brooks-Lyengar algorithm).
4. The nodes broadcast their 'proof' of the aggregation which consists of a message authentication code calculated on the aggregation result in addition to an indication as to which measurements where aggregated.
5. The nodes collect X proofs of their aggregation and store them with the aggregated result.

This approach requires nodes to have a freshness function for the data types, a way to calculate and verify message authentication codes, networking abilities

and neighbor-awareness. Further more this algorithm can be tuned by altering the X, Y parameters and modifying the freshness function. In the prototype this method is partially implemented: the node checks if enough data is available (it could be of the same node) it broadcasts the data (and results) and filters the data and uses the freshness function but is does not collect the proofs of the aggregation as there are very few smartphones in the testbed which does not contain enough nodes for the collection of the nodes.

8.8.4 Aggregation Freshness function

To prevent the aggregation of old and/or wrong data the list of sensor data should be filtered. For this purpose the freshness function is defined.

The following parameters can be used in the freshness function:

- *size* : How much data is available (can be deduced from the data set),
- *src* : Which node is the source of the data,
- *cor* : The correlation of the data,
- *type* : Data type,
- *pos* : Position of the measurement,
- *dTm* : Time between measurements,
- *time* : The time of the measurement (in Android the time since January 1970).

General function
M is the set of available measurements which can be used in aggregation.
type is the type of data to aggregate. Each type can provide parameters: P_{type} and T_{type}. When a parameter is omitted it is assumed to be irrelevant.
P_{type} is a parameter which defines the maximum distance from the center of the aggregation (depended on the type of data).
T_{type} is a parameter which defines the maximum elapsed time between a now and the creation time of a measurement.
pos_{agg} is the position of the aggregation, which is the average location of the nodes that participate in aggregation.
Δd is the time difference between the current time and the creation time of a measurement. (Now - $time_m$).

$$Freshness(type, M) = \{m | m \in M | m \in type | pos_{agg} - P_{type} < pos_m < pos_{agg} + P_{type} | \Delta d_m <= T_{type}\}$$

The parameters that are defined for the current data types in the prototype are:

- Location: $T_{location} = 3$ minutes. $P_{location}$ is omitted. The local location is updated in Android approximately every 30 to 60 seconds when using network locations and the update rate of GPS varies a lot depending on the coverage and device settings. As nodes can move it is important that this parameter is as exact as possible however it is possible that a newer location estimation is not available every minute so a duration of 3 minutes

was chosen for this parameter (with a speed of 25km/h a node can traverse 416m in a minute which is 1,2 km in 3 minutes).

- Composite data (unspecified composite type): use the freshness function with the type of the first registered sensor data.
- 'Double': $T_{location} = 10$ minutes. $P_{location}$ is omitted due to the different sensor types that use this type for storage in combination with a limited communication range. It is assumed that the data is still relevant in the maximum distance a node can traverse in 10 minutes (4.1 km in 10 minutes with a speed of 25km/h).

Other types can easily be added by formulating the data type and defining relevant values for the parameters.

Location aggregation
Android provides locations with a center point and an accuracy factor. The center of a point is in Longitude and Latitude and the accuracy factor of a location is the radius of the center point as one standard deviation (68.2%) in meters. The location of an aggregation is the center point of the sensor locations which where aggregated. If all measured locations overlap then the center location is the center of the common overlapping area. The accuracy takes in account the inaccuracies of the locations by calculating the minimum radius using twice the inaccuracies of the sensors (two standard deviations = 95.4% chance).

To avoid the deviation caused by having a lot of measurements on one side of the node the center is calculated as the average of the min and the max point of locations.

$$Average_{Latitude} = \frac{Latitude_{min}}{2} + \frac{Latitude_{max}}{2}2$$
$$Average_{longitude} = \frac{Longitude_{min}}{2} + \frac{Longitude_{max}}{2}$$

The solution can either be optimized to give a better location estimate or a smaller accuracy value. To optimize for a better location estimate the average of min+ max of distances is used, thereby minimizing the maximum distance from the center to the crossings. Alternately the optimization for a smaller accuracy value is achieved by using minimizing the average distance from a crossing to the center. The prototype is optimized for a better location estimate as it is expected that the deviation caused by having a lot of measurements on one side of the node might pose a problem.

The difference in optimizations is shown in Figures 5 and 6 where the small red circle indicates the estimated center for the two optimizations in the same scenario.

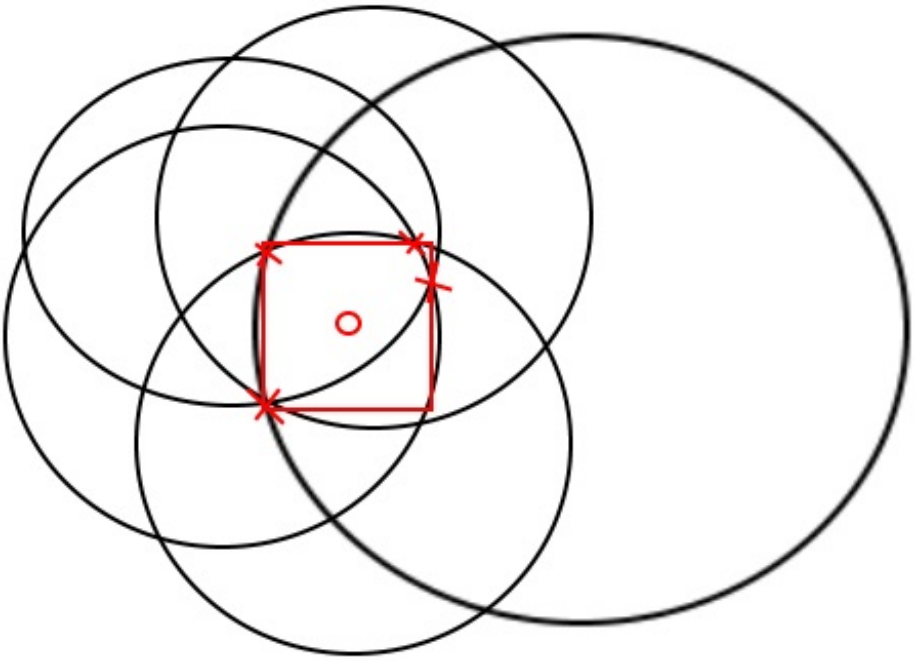


Figure 5: Square with aggregation center, optimized for exactness of the location by using min/min

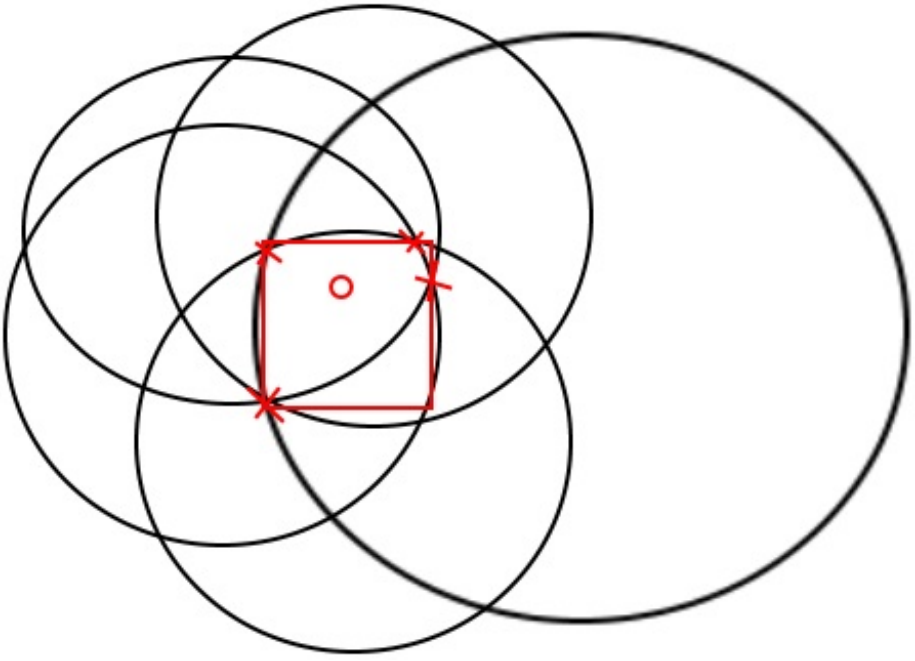


Figure 6: Square with aggregation center, optimized for minimizing the accuracy circle by using an average

9 Implementation

To proof the usability of smartphones in an environmental monitoring scenario a prototype has been developed. This chapter provides the implementation details of the design which is described in chapter 8

The prototype is developed for Android. There are two variants on the implementation, depending on the features used. The full version which supports the use of Bluetooth Low Energy requires Android version 4.3+ whilst a reduced version which cannot connect to sensor nodes but can view data provided by other nodes requires Android version 3.2+.

9.1 Application

The user interface of the prototype (as shown in Figure 7) has 3 tab pages "Data","Settings" and "Location".

"Data" tab page
The data tab page (Figure 7a) presents the measured data in a textual representation to the user. The user can select the type of data they want to see and the application retrieves the information from the database to display it to the user. The window is updated when new data is available of the selected type. The supported types of data to display are

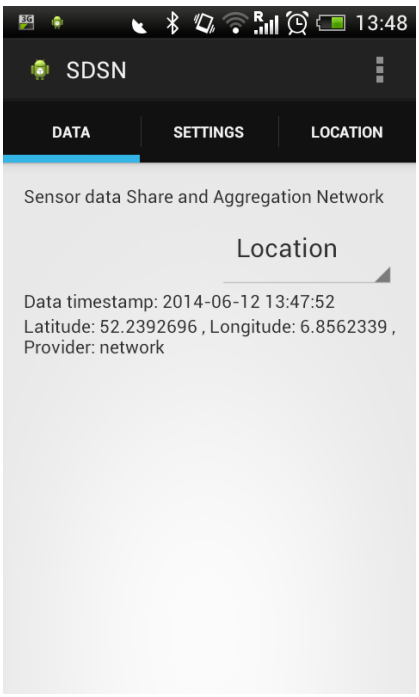
- Undefined - Display all data that is received
- Location - Display location data that is measured or received. Figure 7a is a screenshot in which this representation is selected.
- Double - All separate raw sensor data in a numeric format such as the temperature, humidity and pressure.
- Composite - All aggregated data. This type is a collection of multiple data types which can include other types including other composite types.

"Settings" tab page
The settings tab page (Figure 7b) lets the user edit the most important settings in the application: The upload server address and the instances of the security layer, the communication layer and the aggregation layer. By default the settings will be upload server "10.0.0.112", the security layer will be the Simple Certificate-based security layer, the communication layer will be the Loop-Back communication layer and the default aggregator is the Basic aggregator. The application does not have to be restarted to take updates into account.

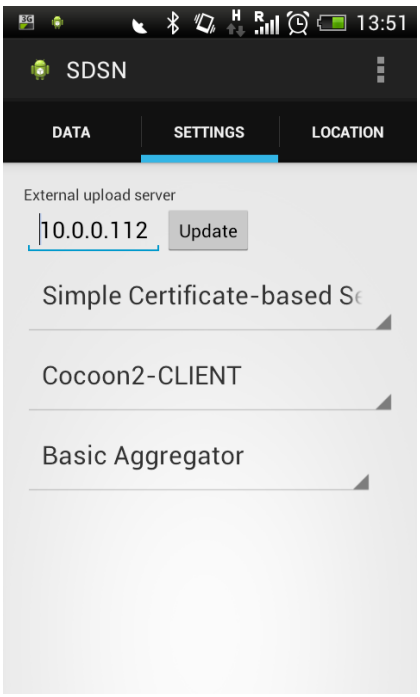
"Location" tab page
The location tab page (Figure 7c) shows the current location of the sensor node and up to 5 most recently known location updates of both the smartphone and the received locations of neighbors. The blue dot indicates the current location of the node and the red markers indicate that the source of the information is the Network-provider. Green markers can also appear, indicating the source was GPS and yellow markers indicate that the source was unknown (might occur

on several devices due to manufacturers implementation details on Android's location services).

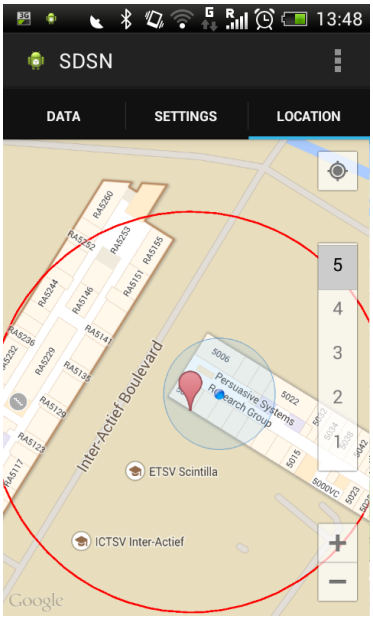
Menu options
The window also has a menu in which the user can initiate predefined processes such as uploading of the (sensor, aggregation, configuration, application logging) data, enabling or disabling the data collection service and clearing the database.



(a) Location data



(b) Settings



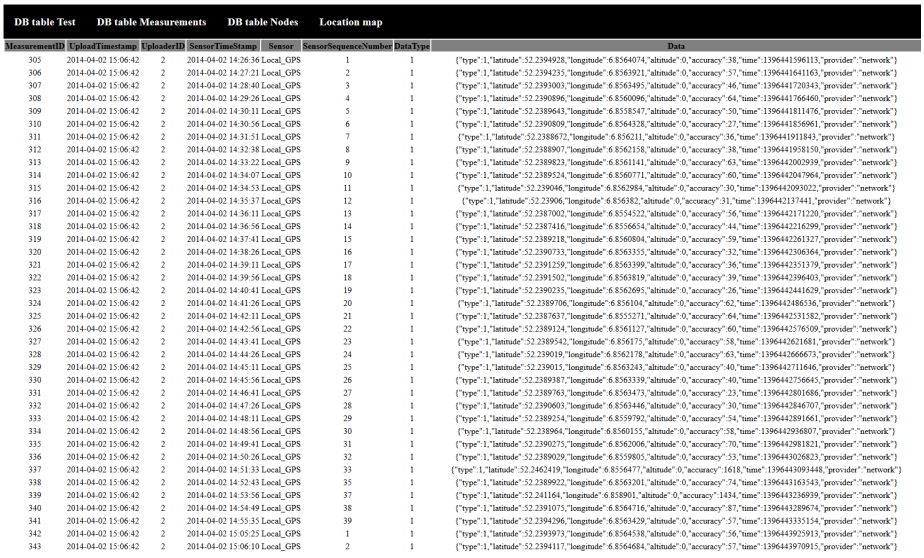
(c) Locations on the map

Figure 7: Screenshots of the prototype

9.2 Web server

For the upload of the data from the smartphone application a web server is used. This webserver accepts uploading of data from the smartphone, acts as a trusted authority to sign certificates from the security-based security layer and provides a simple representation of the stored data in the form of a web page.

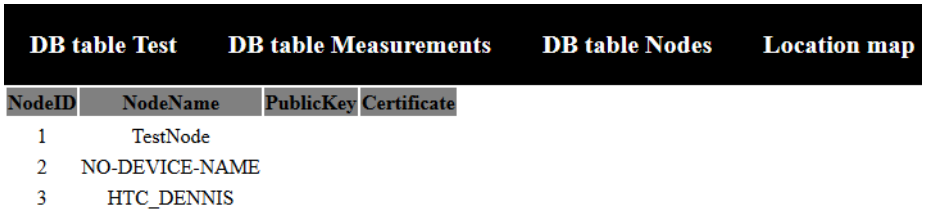
For displaying the data the web server has 3 main elements specialized for the type of data.
The first page is designed to display the measurements in the database. A screenshot of this page is Figure 8.
The second page is designed to display the information on the nodes that are registered at the trusted authority. A screenshot of this page is Figure 9.
The third page is designed to visually represent the locations of the stored data using Google maps. The markers displayed have a tool tip with the timestamp of the data in the format of "YYYY-MM-DD HH:MM:SS". Figure 10 is a screenshot of this page which displays some location estimates where the more accurate estimations are more opaque. The user can navigate the map and even zoom in and out. In the screenshot the circles indicate the accuracy range of the measurement where a smaller circle indicates a more accurate measurement.



DB table Test | DB table Measurements | DB table Nodes | Location map

MeasurementID	UploadTimestamp	UploaderID	SensorTimeStamp	Sensor	SensorSequenceNumber	DataType	Data
305	2014-04-02 15:06:42	2	2014-04-02 14:26:36	Local_GPS	1	1	{"type":1,"latitude":52.2394928,"longitude":6.8564074,"altitude":0,"accuracy":38,"time":1396441596113,"provider":"network"}
306	2014-04-02 15:06:42	2	2014-04-02 14:27:21	Local_GPS	2	1	{"type":1,"latitude":52.2394235,"longitude":6.8563921,"altitude":0,"accuracy":37,"time":1396441641163,"provider":"network"}
307	2014-04-02 15:06:42	2	2014-04-02 14:28:40	Local_GPS	3	1	{"type":1,"latitude":52.2393003,"longitude":6.8563495,"altitude":0,"accuracy":46,"time":1396441720343,"provider":"network"}
308	2014-04-02 15:06:42	2	2014-04-02 14:29:26	Local_GPS	4	1	{"type":1,"latitude":52.2390896,"longitude":6.8560096,"altitude":0,"accuracy":64,"time":1396441766460,"provider":"network"}
309	2014-04-02 15:06:42	2	2014-04-02 14:30:11	Local_GPS	5	1	{"type":1,"latitude":52.2389643,"longitude":6.8558547,"altitude":0,"accuracy":50,"time":1396441811476,"provider":"network"}
310	2014-04-02 15:06:42	2	2014-04-02 14:30:56	Local_GPS	6	1	{"type":1,"latitude":52.2390809,"longitude":6.8564328,"altitude":0,"accuracy":27,"time":1396441856961,"provider":"network"}
311	2014-04-02 15:06:42	2	2014-04-02 14:31:51	Local_GPS	7	1	{"type":1,"latitude":52.2388672,"longitude":6.856211,"altitude":0,"accuracy":36,"time":1396441911843,"provider":"network"}
312	2014-04-02 15:06:42	2	2014-04-02 14:32:38	Local_GPS	8	1	{"type":1,"latitude":52.2388907,"longitude":6.8562158,"altitude":0,"accuracy":38,"time":1396441958150,"provider":"network"}
313	2014-04-02 15:06:42	2	2014-04-02 14:33:22	Local_GPS	9	1	{"type":1,"latitude":52.2389823,"longitude":6.8561141,"altitude":0,"accuracy":63,"time":1396442002939,"provider":"network"}
314	2014-04-02 15:06:42	2	2014-04-02 14:34:07	Local_GPS	10	1	{"type":1,"latitude":52.2389524,"longitude":6.8560771,"altitude":0,"accuracy":60,"time":1396442047964,"provider":"network"}
315	2014-04-02 15:06:42	2	2014-04-02 14:34:53	Local_GPS	11	1	{"type":1,"latitude":52.239046,"longitude":6.8562984,"altitude":0,"accuracy":30,"time":1396442093022,"provider":"network"}
316	2014-04-02 15:06:42	2	2014-04-02 14:35:37	Local_GPS	12	1	{"type":1,"latitude":52.23906,"longitude":6.856382,"altitude":0,"accuracy":31,"time":1396442137441,"provider":"network"}
317	2014-04-02 15:06:42	2	2014-04-02 14:36:11	Local_GPS	13	1	{"type":1,"latitude":52.2387002,"longitude":6.8554522,"altitude":0,"accuracy":56,"time":1396442171220,"provider":"network"}
318	2014-04-02 15:06:42	2	2014-04-02 14:36:56	Local_GPS	14	1	{"type":1,"latitude":52.2387416,"longitude":6.8556654,"altitude":0,"accuracy":44,"time":1396442216299,"provider":"network"}
319	2014-04-02 15:06:42	2	2014-04-02 14:37:41	Local_GPS	15	1	{"type":1,"latitude":52.2389218,"longitude":6.8560804,"altitude":0,"accuracy":59,"time":1396442261327,"provider":"network"}
320	2014-04-02 15:06:42	2	2014-04-02 14:38:26	Local_GPS	16	1	{"type":1,"latitude":52.2390733,"longitude":6.8563355,"altitude":0,"accuracy":32,"time":1396442306364,"provider":"network"}
321	2014-04-02 15:06:42	2	2014-04-02 14:39:11	Local_GPS	17	1	{"type":1,"latitude":52.2391259,"longitude":6.8563399,"altitude":0,"accuracy":36,"time":1396442351379,"provider":"network"}
322	2014-04-02 15:06:42	2	2014-04-02 14:39:56	Local_GPS	18	1	{"type":1,"latitude":52.2391502,"longitude":6.8563819,"altitude":0,"accuracy":39,"time":1396442396403,"provider":"network"}
323	2014-04-02 15:06:42	2	2014-04-02 14:40:41	Local_GPS	19	1	{"type":1,"latitude":52.2390235,"longitude":6.8562695,"altitude":0,"accuracy":26,"time":1396442441629,"provider":"network"}
324	2014-04-02 15:06:42	2	2014-04-02 14:41:26	Local_GPS	20	1	{"type":1,"latitude":52.2389706,"longitude":6.856104,"altitude":0,"accuracy":62,"time":1396442486536,"provider":"network"}
325	2014-04-02 15:06:42	2	2014-04-02 14:42:11	Local_GPS	21	1	{"type":1,"latitude":52.2387637,"longitude":6.8555271,"altitude":0,"accuracy":64,"time":1396442531582,"provider":"network"}
326	2014-04-02 15:06:42	2	2014-04-02 14:42:56	Local_GPS	22	1	{"type":1,"latitude":52.2389124,"longitude":6.8561127,"altitude":0,"accuracy":60,"time":1396442576509,"provider":"network"}
327	2014-04-02 15:06:42	2	2014-04-02 14:43:41	Local_GPS	23	1	{"type":1,"latitude":52.2389542,"longitude":6.856175,"altitude":0,"accuracy":58,"time":1396442621681,"provider":"network"}
328	2014-04-02 15:06:42	2	2014-04-02 14:44:26	Local_GPS	24	1	{"type":1,"latitude":52.239019,"longitude":6.8562178,"altitude":0,"accuracy":63,"time":1396442666673,"provider":"network"}
329	2014-04-02 15:06:42	2	2014-04-02 14:45:11	Local_GPS	25	1	{"type":1,"latitude":52.239015,"longitude":6.8563243,"altitude":0,"accuracy":40,"time":1396442711646,"provider":"network"}
330	2014-04-02 15:06:42	2	2014-04-02 14:45:56	Local_GPS	26	1	{"type":1,"latitude":52.2389387,"longitude":6.8563339,"altitude":0,"accuracy":40,"time":1396442756645,"provider":"network"}
331	2014-04-02 15:06:42	2	2014-04-02 14:46:41	Local_GPS	27	1	{"type":1,"latitude":52.2389763,"longitude":6.8563473,"altitude":0,"accuracy":23,"time":1396442801686,"provider":"network"}
332	2014-04-02 15:06:42	2	2014-04-02 14:47:26	Local_GPS	28	1	{"type":1,"latitude":52.2390603,"longitude":6.8563446,"altitude":0,"accuracy":30,"time":1396442846707,"provider":"network"}
333	2014-04-02 15:06:42	2	2014-04-02 14:48:11	Local_GPS	29	1	{"type":1,"latitude":52.2389254,"longitude":6.8559792,"altitude":0,"accuracy":54,"time":1396442891661,"provider":"network"}
334	2014-04-02 15:06:42	2	2014-04-02 14:48:56	Local_GPS	30	1	{"type":1,"latitude":52.238964,"longitude":6.8560155,"altitude":0,"accuracy":58,"time":1396442936807,"provider":"network"}
335	2014-04-02 15:06:42	2	2014-04-02 14:49:41	Local_GPS	31	1	{"type":1,"latitude":52.2390275,"longitude":6.8562006,"altitude":0,"accuracy":70,"time":1396442981821,"provider":"network"}
336	2014-04-02 15:06:42	2	2014-04-02 14:50:26	Local_GPS	32	1	{"type":1,"latitude":52.2389029,"longitude":6.8559805,"altitude":0,"accuracy":53,"time":1396443026823,"provider":"network"}
337	2014-04-02 15:06:42	2	2014-04-02 14:51:33	Local_GPS	33	1	{"type":1,"latitude":52.2462419,"longitude":6.8556477,"altitude":0,"accuracy":1618,"time":1396443093448,"provider":"network"}
338	2014-04-02 15:06:42	2	2014-04-02 14:52:43	Local_GPS	35	1	{"type":1,"latitude":52.2389922,"longitude":6.8563201,"altitude":0,"accuracy":74,"time":1396443163543,"provider":"network"}
339	2014-04-02 15:06:42	2	2014-04-02 14:53:56	Local_GPS	37	1	{"type":1,"latitude":52.241164,"longitude":6.858901,"altitude":0,"accuracy":1434,"time":1396443236939,"provider":"network"}
340	2014-04-02 15:06:42	2	2014-04-02 14:54:49	Local_GPS	38	1	{"type":1,"latitude":52.2391075,"longitude":6.8564716,"altitude":0,"accuracy":87,"time":1396443289674,"provider":"network"}
341	2014-04-02 15:06:42	2	2014-04-02 14:55:35	Local_GPS	39	1	{"type":1,"latitude":52.2394296,"longitude":6.8563429,"altitude":0,"accuracy":57,"time":1396443335154,"provider":"network"}
342	2014-04-02 15:06:42	2	2014-04-02 15:05:25	Local_GPS	1	1	{"type":1,"latitude":52.2393973,"longitude":6.8564538,"altitude":0,"accuracy":56,"time":1396443925913,"provider":"network"}
343	2014-04-02 15:06:42	2	2014-04-02 15:06:10	Local_GPS	2	1	{"type":1,"latitude":52.2394117,"longitude":6.8564684,"altitude":0,"accuracy":57,"time":1396443970915,"provider":"network"}

Figure 8: Prototype webserver - Measurements



DB table Test | DB table Measurements | DB table Nodes | Location map

NodeID	NodeName	PublicKey	Certificate
1	TestNode		
2	NO-DEVICE-NAME		
3	HTC_DENNIS		

Figure 9: Prototype webserver - Nodes

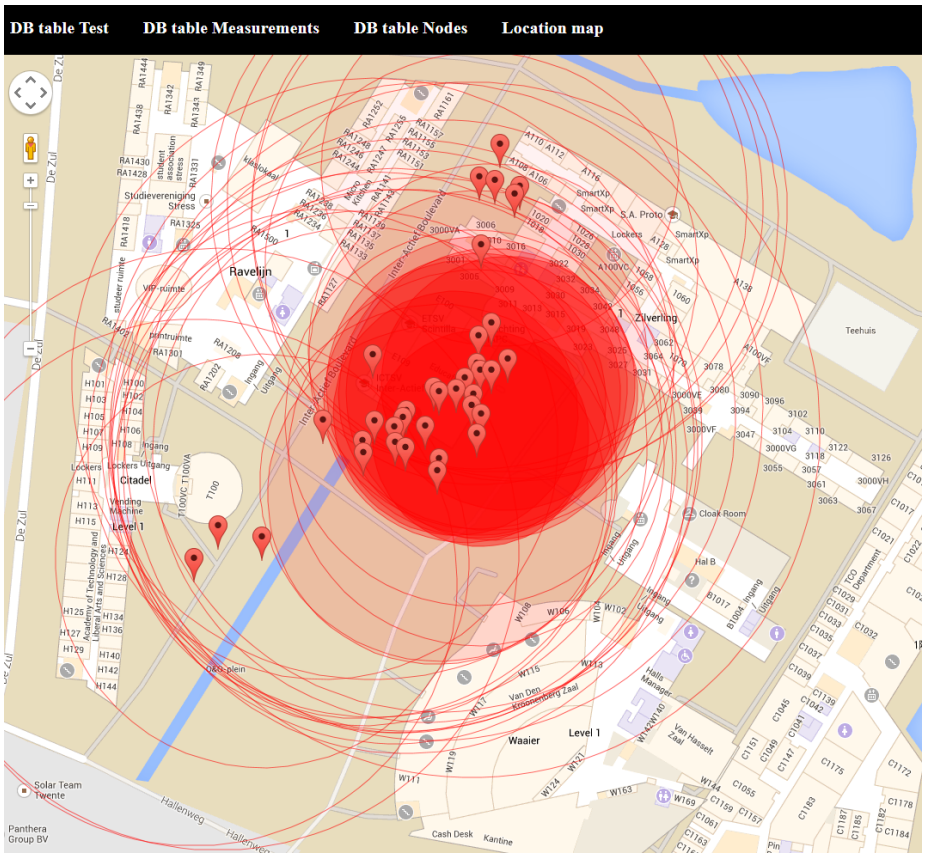


Figure 10: Prototype webserver - Location map

10 Validation

For simulation purposes the ONE simulator (available at `http://www.netlab.tkk.fi/tutkimus/dtn/theone`) [26] version 1.5.1 was used. The Opportunistic Network Environment (ONE) simulator is a simulation environment written in Java used for visualizing mobility and message passing in real-time, generating and emulating node movements and testing various routing algorithms. A few additions and changes were made to the source code to allow GPS locations to be used. Furthermore some load and save mechanics were added for easy testing of aggregation methods.

There are also some practical tests performed to test the prototype. The practical testing was performed using 3 types of smartphones: 1 HTC one V (Android 4.0.3), a few Samsung Galaxy S4 mini's (GT-L9195, Android 4.2.2) and 2 Motorola Moto G (16GB, Android 4.4.2)'s. Of these devices only the Moto G devices support Bluetooth Low energy as the 4.4 update for the S4 mini has not been released at the moment of writing. The sensor node in these tests was the TI CC2541 SensorTag [7] which is a 'dumb' sensor node as it cannot retrieve its own location and cannot store location traces.

10.1 Bluetooth Low energy testing

The simulation of the Bluetooth Low Energy protocol was only tested in a simplified form as time was limited and no simulator was found that supported Bluetooth Low Energy directly. In the One simulator the Bluetooth Low Energy stack was emulated as WiFi with varying ranges. WiFi was chosen as opposed to normal Bluetooth as normal Bluetooth requires pairing and the energy was not measured. The emulated Bluetooth Low Energy was used in the location aggregation tests to retrieve the information of sensor nodes. The rest of this subsection describes the tests of the prototype.

The first Bluetooth Low Energy test was the range test. For this the CC2541 SensorTag device [7] was used and one Moto G device. The test was performed both indoor and outdoors. The SensorTag was placed at around 1 meter height and was put in discoverable mode. Then the smartphone was placed at increasing distances of the SensorTag whilst keeping a line of sight between the smartphone and the SensorTag. At the various placements the range was checked at various ranges by checking if the SensorTag was discoverable by the smartphone and verifying that the data of the SensorTag could be read using the TI BLE SensorTag application (R1.10). The maximum distance measured was around 70 meters indoors and approximately 90 meters outdoors. The difference can be explained by the reflections indoors and the glass window panes in the line of sight.

In the testbed Bluetooth Low Energy was used by the two Moto G devices, which retrieved the data from the sensor node and shared it with nearby phones for aggregation. It appeared not to be possible to connect two devices to the same SensorTag at a time so they were both connected to different SensorTag devices.

10.2 Smartphone Communication-tests

A few simulations were performed in the One simulator with different communication methods: Bluetooth (10m, 100m) Bluetooth Low Energy (10m, 50m, 100m) and WiFi (100m).
Cocoon was simulated as a WiFi network with a epidemic router, in which the nodes could always connect when in range. This is not completely realistic as no hotspot could be in range whilst the nodes are all in client mode. It would also be possible that the nodes are connected to different hotspots in range, thereby not communicating with one another. Lastly nodes can be in communication range but not checking the network (toggling the mode for example) which is also not taken into account in the simulation.
In the simulation the Received Signal Strength indicator (RSSI) was not used for estimation improvement.

The communication range has a lot of influence on the number of aggregations as the inaccuracy increases when the range is increased. The expected cause of the inaccuracy is that the connecting smartphones often follow the same paths and skipping other paths resulting in the sensor node getting only information from one side of its communication range which causes deviation from its real location. Furthermore the number of connections increases when the communi-

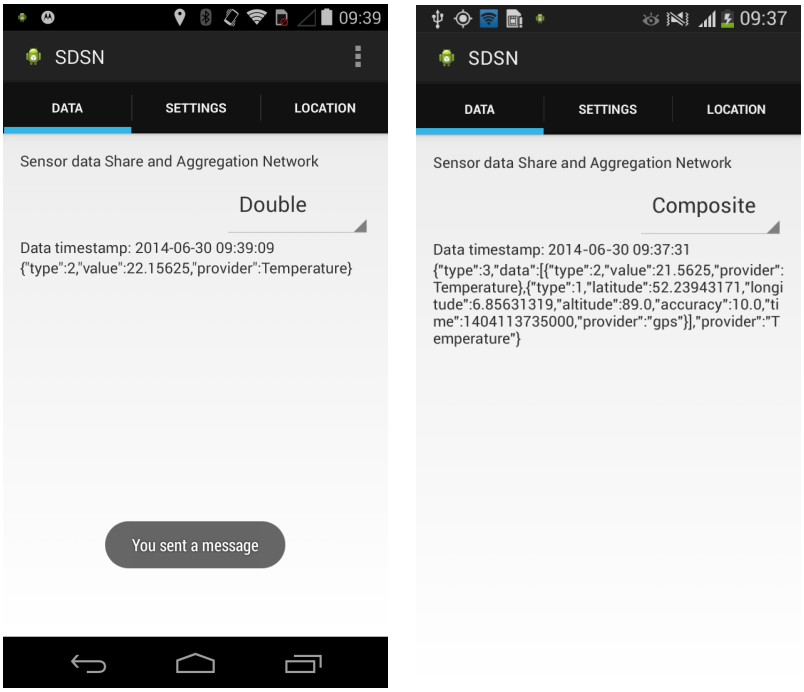
cation range increases as more nodes are in communication range, creating more sensor data to be aggregated with more (due to the increase in range) nearby nodes. In the simulation the packet loss did not pose much of a problem as long as the aggregation- sensor data sharing period was sufficient enough (typically half a minute in the simulation).

Cocoon has also been tested in the testbed where nodes were able to communicate over a range of approximately 100 meters indoors, in this case the network consisted of only 2 nodes (Galaxy s4 mini). The prototype could not use Cocoon on the Moto G devices properly as the hotspot functionality could not always be (reliably) enabled by the application. This appears to be a bug in the firmware and has been listed in this thesis's list of irregularities in Appendix A: Irregularities.

10.3 System test

After testing the separate components it is desired to test the overall system. This subsection describes the test and its results.
The test uses one Moto G, one S4 mini and one SensorTag. The setup is illustrated in figure 11. The Moto G will communicate with the SensorTag via Bluetooth Low Energy and will share its results to the other smartphone using Cocoon. The Moto G is in Cocoon-Client mode and the S4 mini is in Cocoon-Hotspot mode as the Moto G has issues with acting as hotspot as described in Appendix A: Irregularities. It is expected and observed that the S4 mini will display the temperature sensed by the Sensortag. Figures 12a and 12b are screenshots which display the result of the test. The screenshots are of 2 different measurements though, the Moto G measurement is 2 minutes newer than the one displayed on the S4 mini. The temperature shown on the Moto G is $22,16\,^{\circ}\mathrm{C}$ and the temperature shown on the S4 mini is $21,56\,^{\circ}\mathrm{C}$.

Figure 11: System test setup



(a) Moto G (b) S4 mini

Figure 12: System test result

10.4 Aggregation-tests

All aggregation tests were only performed in the simulator due to not having enough devices to reliably provide results.

In the simulator some tests where performed regarding to the aggregation. In these tests the focus was mainly on the aggregation of location data and the effects of the parameters of the freshness function. Recall from section 8.3.1 that the aggregation cycle switches its mode depending on six parameters: Durations of (sharing sensor data, sharing aggregation results, cool down), the number of elements required in the aggregation and the number of broadcasts in the sharing periods. Recall from section 8.8.4 the maximum distance (P_{type}) and the maximum time (T_{type}) parameters.

Parameter effects
It was expected that the duration parameters didn't have much influence on the sensor data available for the aggregation, however during the simulations the effects became apparent. T_{type} was set as 5 minutes, and the aggregation cycle duration took about 5 minutes. The result was that correct sensor data was discarded due to being old (because of the waiting). This combined with a low density and the requirement on the number of elements to be used in aggregation would mean that no aggregation was performed at all. Furthermore the number of participants in aggregation which were able to share the results

with other participating nodes decreased a lot in a more mobile scenario when the durations were long. The reason for this is the movement model as 2 nodes partially overlap their communication range initially but due to movement the communication range will not overlap after a while. The higher the movement the smaller the chance of overlap. Epidemic routers and multi-hop packets can help increase the delivery ratio but decreasing the durations of the aggregation had a better effect in the simulation.

The number of elements required for aggregation had some influence in the simulation, in combination with the density of the network. When the parameter was 1 then all sensor data was aggregated when it was retrieved. A higher density of the network increased the number of aggregations as nearby nodes participated in the aggregation (although not always providing sensor data). When the parameter was higher there were less aggregations as more sensor data was required. A higher density increased the number of available sensor data thereby increasing the number of aggregations. It is recommended that the parameter is set depending on the expected density and the required data updates.

The affects of the P_{type} are not too significant as typically the parameter is set to be larger than the communication ranges plus location inaccuracies. It is expected that in a real testbed this parameter might be problematic as the location inaccuracy increases and the parameter decreases. This effect was apparent in the simulation where the P_{type} parameter was set to lower than the average measurement accuracy. In the test most samples were rejected causing a unreliable location estimation as sometimes no estimation was available at all.

The affects of the T_{type} in the simulation where largely influenced by the duration of the aggregation cycle, the update rate of the data and the retrieval rate of the data. A greater value for the parameter or a higher update rate of the data increases the amount of sensor data available for aggregation which is important for the number of elements parameter. A higher retrieval rate and reducing the aggregation cycle time caused less data to be outdated thereby increasing the amount of data available in the aggregation. In the testbed tweaking this parameter was tricky as nodes could go into conflicting modes thereby not communicating with one another (cooldown and sharing sensor data- mode for example). Furthermore the mobility and the inter-smartphone communication range where important to the number of aggregations as more-mobile nodes would often get outside the communication range, especially with lower communication ranges.

Location aggregation accuracy
The accuracy of the location aggregation was also tested in the simulator. It is preferred that the deviation is as minimal as possible depending on the locations of the registered nodes. It is expected that as the number of traces increases (and using the center of min-max) the deviation decreases like exponential decay. The optimal case is expected to be having 4 points which are at the maximum distance of the communication range and are at different quarters at maximum

communication distance (which should be equal to the node's communication range) as depicted in figure 13. When the nodes are placed in the optimal location then the estimated location will be exactly at the correct positions.

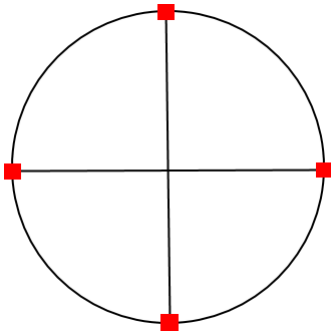


Figure 13: Optimal trace locations
The traces are indicated as red dots

Figures 14 and 15 are the result of the aggregation accuracy test. In these tests up to 100 randomized locations near a node (in communication range) were aggregated to estimate the node location. These tests were performed 200 times for each combination of range and number of locations, combining the results by taking the minimum, maximum and average. Interesting is the dip in using 2 nodes, which can be explained by probabilistic theories:

- Take 2 random points in the communication circle. On average they are at half the communication radius from the center of the circle.
- Divide the circle into 4 quarters. The aggregation will not improve (or only slightly) if the two nodes are in the same quarter.
- The chance for one of the two nodes to be in a specified quarter is $\frac{1}{4} * 2$. The chance that they are both in the same quarter is $\frac{1}{4} * \frac{1}{4} * 4 = \frac{1}{4}$, thus having a chance of $\frac{3}{4}$ to improve the aggregation as the minimum and maximum are used.
- When a third point is added it increases only has a limited extra benefit as the chance that the third is in a different quarter as the other nodes is only $\frac{1}{4} * \frac{1}{4} * \frac{1}{4} * 4 = \frac{1}{16}$.

Figure 14 shows the difference in average deviation from the expected center, which shows that using the min/max in the aggregation performs better than taking the average. It is expected that using the average in scenarios with points of interest (which are not at the center position) will increase the deviation even more.

In the real world traces are not taken randomly as people move on roads and wont be able to be in certain places such as obstructed areas and will often follow the roads. To emulate this behavior two types of tests were performed: one where users followed the road and the other test focuses on points of interest (but traces can be in all areas).

The result of some scenarios of the simulator (where users followed a road) is shown in figures 17and 16. The blue dots indicate the node of which the location

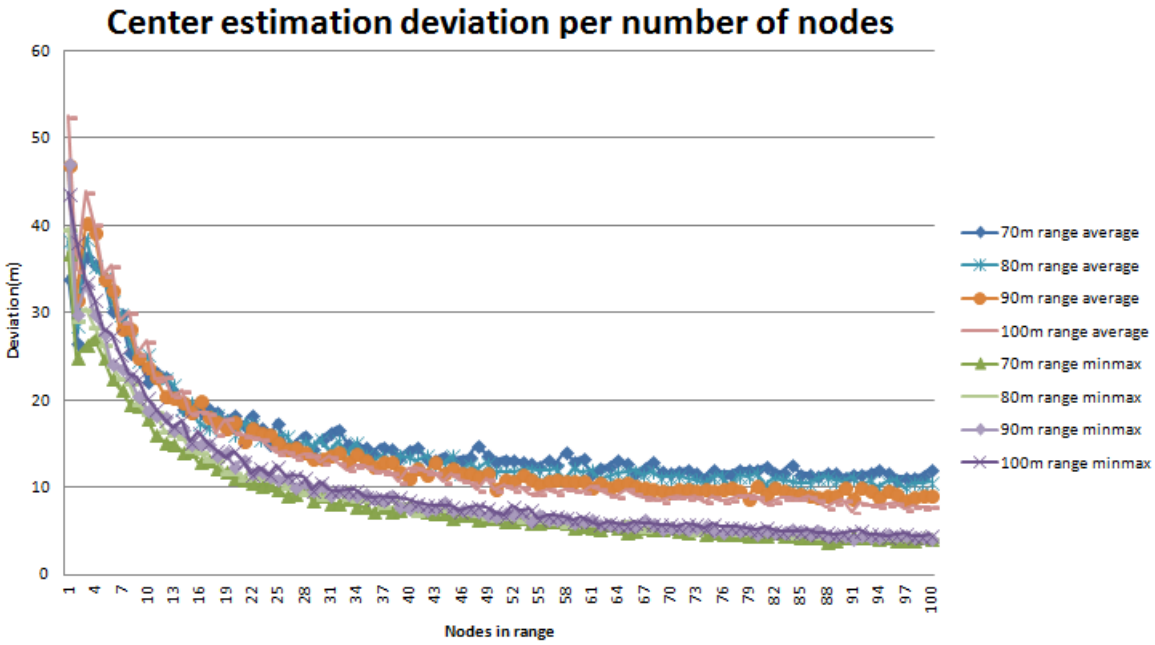


Figure 14: Location Aggregation deviation from exact location
MinMax and Average estimations
Average of 200 tests

should be estimated, the black dots are the estimated locations and the pink dots are recorded traces. Figure 16 shows the result of two estimations with a partially overlapping data set, the dotted lines have been added to indicate to which node the estimations belong.

The accuracy test was not performed in the testbed as it is hard to get enough samples to reliably estimate the average performance. To generate samples for the test one has to move around with a sensor node, in a specific area of interest whilst recording all location traces plus the users exact location. The user's real location is required as the location estimation of smartphones also has inaccuracies and the extra inaccuracy is relevant for the accuracy test.

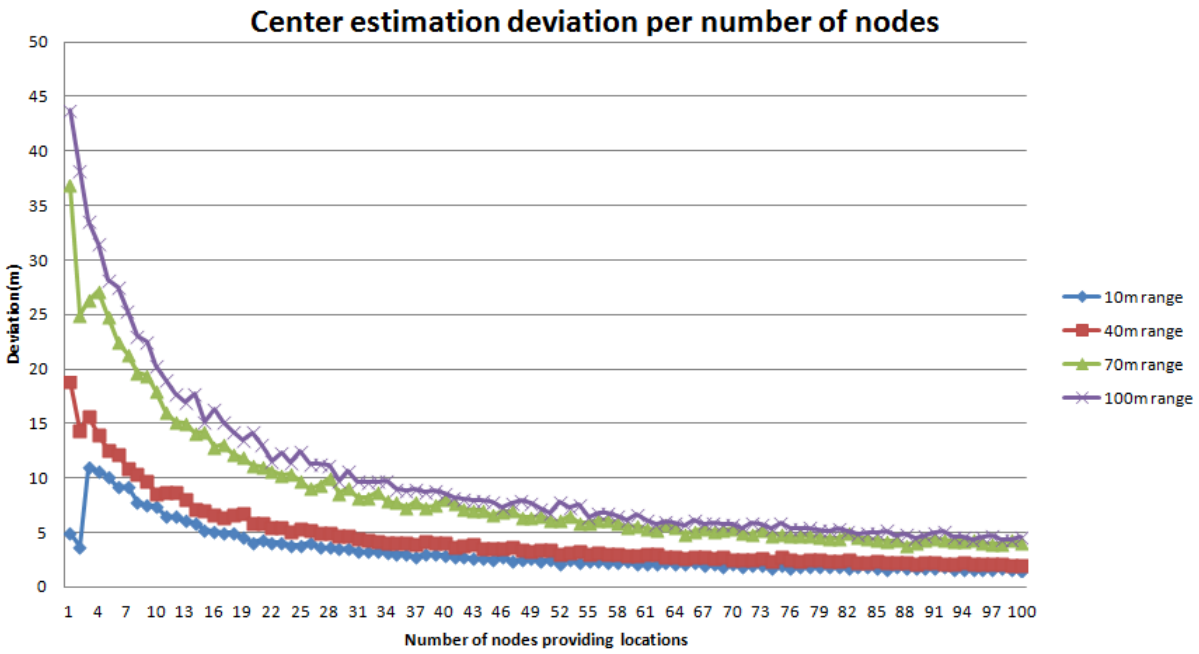


Figure 15: Location Aggregation deviation from exact location
MinMax aggregation
Average of 200 tests

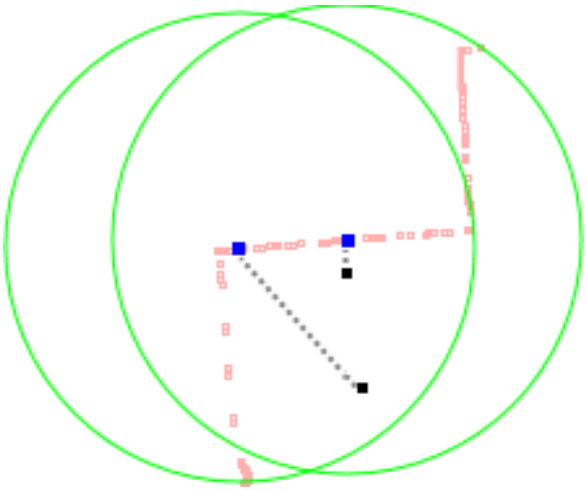
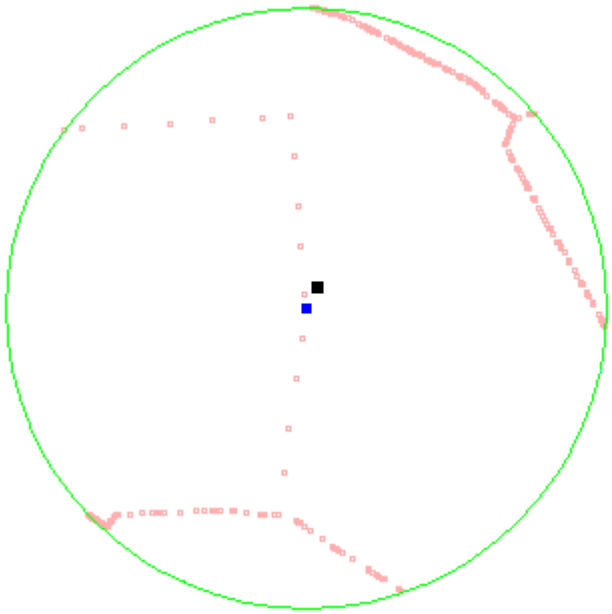
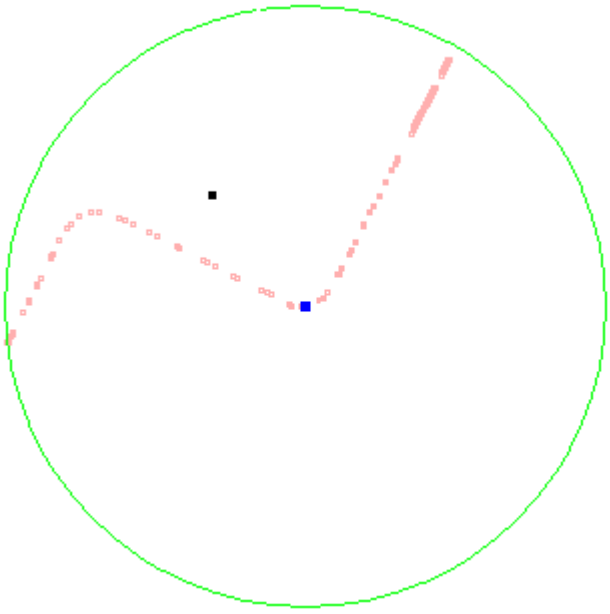


Figure 16: Location estimation, combination of 2 nodes, using 150m communication range
Deviations are 75m and 5m



(a) 13m deviation



(b) 67m deviation

Figure 17: Location estimation, 150m communication range

11 Discussion

When something new is proposed there are always points of discussion and improvement, as is the case in this thesis. This chapter discusses some issues of the system whilst appendix A lists the irregularities encountered during the thesis period.

11.1 Privacy of the system

There are some privacy issues in the prototype which allows tracing users. For example the location data will let an attacker trace where the uploader was at a time (approximately as it had to take part in the aggregation to collect this information). This is even more accurate when raw sensor data is uploaded (instead of aggregated data). It is not possible to trace the node back to a specific smartphone or user however if the attacker knows the node of a user then that user can be traced on uploading of its data. One of the solutions is to anonymize nodes on upload, thereby making it harder to link nodes to smartphones.

11.2 Quality of testing

The testing of the system was pretty limited due to the novelty of Bluetooth Low Energy and the lack of devices supporting Bluetooth Low Energy. Furthermore there was only one type of sensor available (TI SensorTag) for testing which supports Bluetooth Low Energy which meant that the prototype could only be tested in a limited environment. No suitable simulator was found to fully emulate the prototype/design as Bluetooth Low Energy integration, so only parts could be simulated. The testing was done mostly indoors which degrades the accuracy of the location determination of smartphones and it is expected that the location estimation will perform better outdoors. Furthermore the testing was mostly performed with a line of sight connection to other nodes which is not probably in a real scenario as users often carry their smartphones in their pockets instead of carrying the smartphone in hand.

11.3 Feasibility of real use

There are several factors which influence the feasibility of real usage of the system. The most challenging factors for real use are the ease of usage and adoption, the power consumption and the participation of users.

Ease of usage and adaption
An important aspect in the usability of the system is the ease of usage. The ease of usage can be categorized into two phases: the initial ease of usage and the continued ease of usage. The initial ease of usage is the required effort to setup the application and learn how to use it meaningfully and the continued ease of usage is the required effort to keep using the application after learning how to use it.
An important factor for the initial ease of usage is the availability of the application. It should be clear to the users that they can or cannot use the application on their phone. In Android applications can be places in the Play Store, where

users can then download the application directly. It is possible to specify requirements on the smartphone in the Play Store, such as minimum Android version, the necessity of Bluetooth Low Energy or dependencies on map services. Furthermore the user has to, in the current situation, initially connect to a server to get a signed certificate in order to let the smartphone communicate with other smartphones. It should be clear to the user that the initial connectivity is required, and how the user should perform this task. The learning curve of using the application is expected to be short as the application is relatively simple for normal users as the user interface only provides a representation of the data and there are only few basic settings that can be selected.
The second aspect for the initial usage is the availability of the required technologies, which in this case are the requirement of having Bluetooth Low Energy, which is expected by Bluetooth SIG to be available in more than 90 percent of the Bluetooth enabled devices by 2018 [3], and the availability of Android version 4.3+ which is available for nearly all newly sold Android devices.

The continued ease of usage is largely impacted by the interference of the application on the normal smartphone usage of the user. The interference can categorized as:

1. Interfering technologies: Using a technology renders other desired techniques unusable. THis is the case when using Cocoon as it is impossible to use (normal) WiFi at the same time or in Bluetooth where it may be impossible to use Bluetooth for other appliances.

2. Power consumption: Smartphones only have a limited power supply, which is drained faster by using the application. The amount of extra power used depends on which features are used and how often they are used. For example using the GPS to estimate the location requires a lot more energy than using network based location estimates. Scanning of Bluetooth and WiFi channels also take a lot more energy compared to the idle modes of the techniques.

3. Other resource usage: The application requires memory and processing time on the device, reducing the amount available for other applications. Furthermore it requires storage space to store logging files and results.

Power consumption
Using the application can have a large impact on the battery-time of smartphones depending on the settings used.
On using the GPS the battery drains a lot faster compared to network-only mode however it might be advantageous to use the GPS as it increases the accuracy.
The communication modes of the available techniques also influence the battery usage as for example Cocoon client mode consumes far less energy than Cocoon hotspot mode.
Furthermore Bluetooth typically uses less energy than WiFi-based techniques [14].
The use of Bluetooth Low Energy has relatively little impact as it is designed to keep the energy consumption minimal.
Lastly the display on the smartphone consumes power. The application may

run in the background so the screen may be switched off or other applications can be ran in the foreground.

Incentives to participate
To improve the number of participants it is important to understand the reasons why people participate. In [10] S. Aflaki lists the incentives and motivations commonly used for opportunistic sensing based environmental monitoring projects. The motivations for participation in environmental monitoring projects can be categorized into:

- Supporting scientific research
- Reputation
- Solving problems
- Altruism, unselfishness participation
- Socializing
- Financial reasons (being paid to participate or a discounts)

Some popular incentives to increase the motivations for participation are:

- Game - Creating a game around the sensing or project.
- Feedback - Feedback about the situation, optionally with displayed 'improvement' achieved by the project.
- Socializing - Performing a task together/ ranking among others.
- Rewards (financial, free calls, free stuff) - Various rewards for participation.

Depending on the motivation for the participation participants have different expectancies of the project, which is mostly related to the combination of incentives and motivations for participation. For example someone who participates for 'supporting scientific research' would probably want feedback on how they helped. Participants that participate for altruistic reasons may not want special rewards but someone who participates for financial reasons expects the rewards.

11.4 On the communications and portability

The prototype is designed for Android which means that it doesn't run directly on other mobile operating systems. It is however possible to port the application in reasonable time to different operating systems as the code is written mostly in plain Java which most operating systems support. Cocoon however is far harder to port to different operating systems as it uses a lot of lower level - Android-specific functions. Bluetooth is available on most platforms, however insecure pairing mode is only available for Android devices. The external upload can easily be ported to different operating systems.
IBeacons support might be interesting to add to the system if it becomes more popular in the future as it can have a reasonable range whilst using Bluetooth

based techniques. It is not possible to add this technology yet to the system as it is only supported only by Apple products.
WiFi-Direct could be added as an additional communication technique as it is incorporated in most newer smartphones making it more and more available. An added benefit is that the smartphone can use WiFi whilst using WiFi Direct (depending on manufacturers implementation). Sadly the Android API is not yet very mature on the subject, making it hard to use this technique at the moment.

12 Conclusion

This chapter concludes the thesis by shortly repeating the answers to the (sub) research questions and summarizing the thesis results.

To provide an answer to the main research question four sub research questions where defined which help answer the main question:

1. *What is opportunistic sensing and what are its challenges?*
This sub-question is covered in chapter 4 and can be summarized as the collaboration of (typically mobile) nodes to achieve a common goal by working together towards a common goal. The operation of these nodes should still be possible in the absence of a stable and permanent communication network. The related relevant challenges are: The participation of users, Security challenges, trust in the system and ensuring privacy.

2. *What is environmental sensing and what are its challenges?*
This sub question is covered in chapter 3 and can be summarized as "sensing the environment", measuring different characteristics with different types of sensors for a specific purpose such as measuring the ecology (temperature, humidity), air quality (CO_2, fine dust) or water quality (detection of pollution). Environmental sensing is often done using networks of sensor nodes which sense the environment and send their data to a central point, where processing happens and the data is send to a server for further analysis.

The challenges in environmental sensing can be divided into two categories: Challenges related to using sensor nodes and the processing/using the collected data.

The standard challenges related to using sensor nodes are mostly related to the fact that typical sensor nodes are resource constrained. The most important challenge is the lifetime of the sensor nodes, which is mostly depended on the energy available and energy. Furthermore there are challenges related to the communication (duty cycle, routing of data, etc), the sensing of data and the localization of nodes.

The common challenges for processing/using the collected data are: The correlation of collected data, the interpretation and extraction of data, privacy of the data (providers) and the security and authenticity of the data.

3. *What capabilities do smartphones have to offer for environmental sensing, and how feasible are their uses for environmental sensing?*

Recall from chapter 3 and the previous research question the challenges in environmental sensing in relation to sensor nodes: Lifetime of the sensor node, challenges related to the communication, challenges related to sensing and the localization of nodes.

Using smartphones as part of an environmental sensing network can be the solution to the challenges. The challenge of the lifetime of a sensor node is reduced as users of smartphones often recharge the batteries of their phones. It is still important to minimize the power consumption to reduce the intrusion in the normal usage of smartphones as users will want to use their smartphone for dif-

ferent purposes, not only environmental monitoring. Smartphones have several communication techniques available which can be used in communicating with sensor nodes or other smartphones (as listed in chapter 7) such as using Bluetooth and Bluetooth Low Energy however sensor nodes may also use techniques which are not supported by smartphones like Zigbee. Furthermore the typical routing issues of sensor nodes can still be relevant when using smartphones in sensor networks depending on the communication techniques. Smartphones do have some sensing capabilities but are typically not designed with sensors that measure their environment (except light and temperature). Smartphones have capabilities to determine their location exactly using either GPS or network information.

There are also some major challenges of using smartphones for environmental sensing (in the context of a smartphone based environmental sensing network): The participation of users (both initial and continued participation), the technology availability and the interference with normal usage of the smartphone.

Smartphones solve some of the problems in sensor networks but are not suitable to replace all sensor nodes due to the lack of relevant sensors. It is recommended to use a combination of sensor nodes (which perform the sensing) and smartphones (which collect and share the information) in an environmental sensing network.

4. *What are the challenges in inter-smartphone communication in the context of an opportunistic network?*
This sub question is covered in chapters 5 and 7, focusing on answering this question in the specific case of the Android operating system. The major issue in inter-smartphone communication (in the case of Android) is the fragmentation of Android. There are a lot of Android devices which differ in device-capabilities, manufacturer specific implementations and versions of Android. The effect of the fragmentation is that not all techniques are available for all devices. There are several techniques available for inter-smartphone communication which can be used, but are not all equally suited. Chapter 7 provides a comparison of the techniques to determine their suitability. For this purpose the following aspects are compared (which are also challenges in the smartphone to smartphone communication): The transmission range, the availability of the technique, the ease of access and the capacity of the technique.

Using the sub research questions the main research question can be answered: *"How can smartphones be used in environmental sensing, in the context of a city-wide project, and what are the challenges to overcome?"*

Smartphones can be used in numerous ways in environmental sensing, performing different tasks. Smartphones can act as powerful mobile sensor nodes of which the battery is often recharged and has good processing power and a range of communication techniques; They can act as temporary infrastructure providing coverage to other nodes (for example when using Cocoon) or they can act as a simple information provider to users, displaying relevant data on demand. Smartphones have several communication capabilities which can be used in an environmental monitoring network, all with different pro's and con's

in relation to different aspects of communication. The most promising techniques for transmitting data between smartphones in environmental monitoring are Bluetooth, Bluetooth Low Energy and WiFi.

In order to demonstrate the usability of smartphones in environmental sensing a prototype was designed, implemented and tested in a testbed. Parts of the design have also been tested in a simulator.
The prototype smartphone application was able to:

- Communicate to the sensor node via Bluetooth Low Energy.
- Communicate with nearby smartphones via Cocoon.
- Aggregate received sensor data and share the sensor- and aggregated data.
- Visualize the aggregated sensor data (in the application).
- Upload the data to the external server.

Furthermore a website was developed to support the upload of data and visualizing the uploaded results.

The prototype demonstrated the feasibility of using smartphones in environmental sensing as it was able to act as a communication link between sensor nodes and the external server. In addition the smartphones provided additional functionality to the system as they were able to communicate with different nodes which might otherwise not be connected to the network, they add location awareness to the network as the smartphones can estimate the location of the sensor nodes and they provide a limited amount of additional redundancy as multiple different smartphones may communicate with the node instead of a single communication link between two sensor nodes. There are also some downsides such as the price of smartphones and the required participation of a sufficient of users and the unreliable collection of data as a node may have a lot of communications or very few depending on the number of users nearby.

The simulations performed confirmed the feasibility of using the smartphones on a larger scale than tested in the testbed.

13 Future Work

This section lists some of the possible future work to be performed in order to improve the prototype and the quality of testing.

Larger scale testing
The tests that have been performed were only in a small setting, having only two smartphones which had Bluetooth Low Energy capabilities. It is proposed that larger scale tests are performed with multiple smartphones connected to multiple Bluetooth Low energy devices in order to test the behavior for more realistic scenarios.

More simulations
To improve the quality of testing more simulations should be run, using different simulators and different movement models. This should improve the trust in the performed simulations and indicate (undiscovered) weak points of the system. Furthermore the current simulations did not take night-time into account, in which it is expected that only smartphones are in communication range with the sensor nodes.

Different types of sensors
The prototype currently only supports one type of sensor node which should be expended to cover different types of nodes with different sensor data types.

Real deployment
The prototype should be tested in a real deployment outdoors instead of indoors. The real deployment should also contain more sensors and smartphones which can communicate via Bluetooth Low Energy.

Different operating systems
The prototype is currently only available for Android. By increasing the availability of the application by expanding it to different platforms more people can participate in the network, increasing the coverage of the system.

Newer Cocoon versions
It is also recommend that more tests are run with newer versions of Cocoon as currently only an older experimental version is used. Different versions of Cocoon may have a large impact on the functioning of the system as sharing of the sensor data is strongly dependent on the availability of neighbor nodes.

Acknowledgments

I would like to thank Okan Türkes for providing me with an early version of Cocoon and assisting with its usage.
I would also like to thank my supervisor Hans Scholten for providing writing tips and early proof-reading.

References

[1] http://source.android.com/source/index.html.

[2] Android-dashboard. http://developer.android.com/about/dashboards/index.html. Accessed at: 8-1-2014.

[3] Mobile telephony market. http://www.bluetooth.com/Pages/Mobile-Telephony-Market.aspx.

[4] Node+ sensor platform. Available at http://shop.variableinc.com/products/node.

[5] Punch Through Design - Light Blue Bean. Available at http://www.punchthrough.com/products.

[6] Stack Overflow - WiFi Direct (Android 4.0) with multiple (3+) devices. http://stackoverflow.com/questions/11251610/wifi-direct-android-4-0-with-multiple-3-devices. Accessed at 10-1-2014.

[7] Texas instruments CC2541 sensor tag development kit.

[8] Industry leaders announce open platform for mobile devices. http://www.openhandsetalliance.com/press_110507.html, nov 2007.

[9] Bluetooth core specification version 4.0 ready to roll. http://www.bluetooth.com/Pages/Press-Releases-Detail.aspx?ItemID=101, apr 2010.

[10] S. Aflaki. On analysis of incentive tools for sensing based environmental monitoring applications. 2014.

[11] Hani Alzaid. *Secure data aggregation in wireless sensor networks*. PhD thesis, Queensland University of Technology, 2011.

[12] S. Aram, A. Troiano, and E. Pasero. Environment sensing using smartphone. In *Sensors Applications Symposium (SAS), 2012 IEEE*, pages 1–4, Feb 2012.

[13] Siamak Aram, Amedeo Troiano, Francesco Rugiano, and Eros Pasero. Low power and bluetooth-based wireless sensor network for environmental sensing using smartphones. In *Artificial Intelligence Applications and Innovations*, pages 332–340. Springer, 2012.

[14] Rahul Balani. Energy consumption analysis for bluetooth, wifi and cellular networks. *Online]. http://nesl. ee. ucla. edu/fw/documents/reports/2007/PowerAnalysis. pdf*, 2007.

[15] Erik-Oliver Blaß, Joachim Wilke, and Martina Zitterbart. Relaxed authenticity for data aggregation in wireless sensor networks. In *Proceedings of the 4th international conference on Security and privacy in communication netowrks*, page 4. ACM, 2008.

[16] R.R. Brooks and S.S. Iyengar. Robust distributed computing and sensing algorithm. *Computer*, 29(6):53–60, Jun 1996.

[17] Andrew T. Campbell, Shane B. Eisenman, Nicholas D. Lane, Emiliano Miluzzo, and Ronald A. Peterson. People-centric urban sensing. In *Proceedings of the 2Nd Annual International Workshop on Wireless Internet*, WICON '06, New York, NY, USA, 2006. ACM.

[18] Georgios Chatzimilioudis, Andreas Konstantinidis, Christos Laoudias, and Demetrios Zeinalipour-Yazti. Crowdsourcing with smartphones. *Internet Computing, IEEE*, 16(5):36–44, 2012.

[19] Yohan Chon, Nicholas D Lane, Fan Li, Hojung Cha, and Feng Zhao. Automatically characterizing places with opportunistic crowdsensing using smartphones. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, pages 481–490. ACM, 2012.

[20] Wenliang Du, Jing Deng, Yunghsiang S Han, and Pramod K Varshney. A witness-based approach for data fusion assurance in wireless sensor networks. In *Global Telecommunications Conference, 2003. GLOBECOM'03. IEEE*, volume 3, pages 1435–1439. IEEE, 2003.

[21] Jon Fingas. Android climbed to 79 percent of smartphone market share in 2013, but its growth has slowed. `http://www.engadget.com/2014/01/29/strategy-analytics-2013-smartphone-share`, jan 2014.

[22] David Hasenfratz, Olga Saukh, Silvan Sturzenegger, and Lothar Thiele. Participatory air pollution monitoring using smartphones. *Mobile Sensing*, 2012.

[23] Bret Hull, Vladimir Bychkovsky, Yang Zhang, Kevin Chen, Michel Goraczko, Allen Miu, Eugene Shih, Hari Balakrishnan, and Samuel Madden. Cartel: A distributed mobile sensor computing system. In *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems*, SenSys '06, pages 125–138, New York, NY, USA, 2006. ACM.

[24] Chalermek Intanagonwiwat, Ramesh Govindan, and Deborah Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*, MobiCom '00, pages 56–67, New York, NY, USA, 2000. ACM.

[25] Salil S Kanhere. Participatory sensing: Crowdsourcing data from mobile smartphones in urban spaces. In *Mobile Data Management (MDM), 2011 12th IEEE International Conference on*, volume 2, pages 3–6. IEEE, 2011.

[26] Ari Keränen, Jörg Ott, and Teemu Kärkkäinen. The ONE Simulator for DTN Protocol Evaluation. In *SIMUTools '09: Proceedings of the 2nd International Conference on Simulation Tools and Techniques*, New York, NY, USA, 2009. ICST.

[27] Joanna Kulik, Wendi Heinzelman, and Hari Balakrishnan. Negotiation-based protocols for disseminating information in wireless sensor networks. *Wireless networks*, 8(2/3):169–185, 2002.

[28] Martin Leopold, Mads Bondo Dydensborg, and Philippe Bonnet. Bluetooth and sensor networks: A reality check. In *Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 103–113. ACM, 2003.

[29] Bartosz Przydatek, Dawn Song, and Adrian Perrig. Sia: Secure information aggregation in sensor networks. In *Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 255–265. ACM, 2003.

[30] Ramesh Rajagopalan and Pramod K Varshney. Data aggregation techniques in sensor networks: A survey. 2006.

[31] Hans Scholten and Pascal Bakker. Opportunistic sensing in train safety systems. *International Journal on Advances in Networks and Services*, 4(3-4):353–362, 2011.

[32] Bluetooth SIG. About bluetooth low energy model. `http://www.bluetooth.com/Pages/low-energy-tech-info.aspx`.

[33] Chris Smith. Immensely popular iOS app FireChat brings Internet-less chat to Android. April 2014.

[34] Matthias Stevens and Ellie DHondt. Crowdsourcing of pollution data using smartphones. In *Workshop on Ubiquitous Crowdsourcing*, 2010.

[35] Okan Turkes. Cocoon software. Pervasive Systems Group - University of Twente, not yet published.

[36] Hamilton Turner and Jules White. Verification and validation of smartphone sensor networks. In Nalini Venkatasubramanian, Vladimir Getov, and Stephan Steglich, editors, *Mobile Wireless Middleware, Operating Systems, and Applications*, volume 93 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 233–247. Springer Berlin Heidelberg, 2012.

[37] Debao Xiao, Meijuan Wei, and Ying Zhou. Secure-spin: Secure sensor protocol for information via negotiation for wireless sensor networks. In *Industrial Electronics and Applications, 2006 1ST IEEE Conference on*, pages 1–4. IEEE, 2006.

Appendix A: Irregularities

During the thesis period some irregularities were observed:

- Key-pair generation whilst debugging the application takes a long time on the Android smartphones.
- Devices react differently on Bluetooth connections: for example the HTC one V spams debug messages when Bluetooth is used while the Nexus 7 shows warning-debug messages when the connection is used (getBluetoothService() called with no BluetoothManagerCallback)

- Not all devices support WiFi-Direct even though the API version supports it (HTC one V). Also some devices have Bluetooth 4.0 but their OS version doesn't support it yet.
- The web server needs write-rights to the upload directory as it has to store and save the uploaded files.
- Different Android versions sometimes require different functions to be used (for example in using Bluetooth).
- In Android there is a hard limit of 4 unique notification subscriptions for a connection in Bluetooth Low Energy, limiting the number of characteristics which are automatically updated.
- Some Android Hot-Spots will not be enabled or disabled by software, for example in the Moto G devices. Other smartphones may require a SIM-card to be installed in order to enable Hot-Spot mode.
- The accuracy of location updates in Android vary a lot depending on the device, for example the HTC one V receives location updates less often and typically less accurate compared to the Moto G devices. Furthermore newer Android devices have the option to select which location mode is enabled: High accuracy which uses GPS,Wi-Fi and mobile networks, Battery saving which uses Wi-Fi and mobile networks and lastly Device only which uses only GPS to determine the location. In the experiments in this thesis the High accuracy mode is used.

Appendix B: Bluetooth Low Energy Implementation details

For the communication between the smartphone and the sensor board Bluetooth Low Energy is used. This appendix provides the implementation details on the usage of Bluetooth Low Energy in the prototype. The terminology used here is described in chapters 8.6 and 7.1.

The GATT service contains 2 out of 3 characteristics: SensorData (Mandatory) and either LocationExact or LocationSet. The sensor data characteristic should be in the same fashion as between smartphones (Chapter 8.4) to ensure simplicity.

To identify GATT services, characteristics and descriptors in Bluetooth and Bluetooth Low Energy universally unique identifiers (UUIDs) are used. The following UUIDs are proposed (but not used in the prototype) for the communication with the sensor node:

- ServiceName [06564499-28ba-3c92-8105-b4452f616c0e] - UUID to identify the GATT-service.
- LocationExact [f257d837-b1fd-3762-a9dd-dbfab9f22eec] - UUID to identify the GATT-characteristic which indicates an exact location.
- LocationSet [a9eee3eb-13e1-3233-961c-7c992f43cd24] - UUID to identify the GATT-characteristic which indicates a set of recorded locations.
- SensorData [58a77abd-1586-3139-8d34-07518b720896] - UUID to identify the GATT-characteristic Sensor data.

The SensorTag is used as sensor node which does not support the use of custom services and characteristics. The result of this is that the services and characteristics of the SensorTag should be used. The relevant services and characteristics of the SensorTag are:

- Service Temperature[f000aa00-0451-4000-b000-000000000000]
- Service Humidity [f000aa20-0451-4000-b000-000000000000]
- Service Barometer [f000aa40-0451-4000-b000-000000000000]
- Characteristic Temperature Data [f000aa01-0451-4000-b000-000000000000]
- Characteristic Humidity Data [f000aa21-0451-4000-b000-000000000000]
- Characteristic Barometer Data [f000aa41-0451-4000-b000-000000000000]

Only the service temperature and characteristic temperature data were used in the prototype to provide data. Of the 'Characteristic Temperature Data' characteristic only the environmental temperature is calculated and used. The other listed characteristics are partially implemented and not used.

Appendix C: Inter-Layer packet layout

For the communication between the different layers in the prototype message packets are required. Between each layer (communication/security/aggregation) there can be several different packet layouts as long as all nodes use the same inter-layer implementation. The specified packet formats (as used in the prototype) are:

Aggregation layer packet:

Sensor data type [1b]	Sensor data [variable]

Security layer packet:

Source [15b]	Sequence number [1b]	Aggregation packet length [4b]	Aggregation packet [variable]	Signature of packet [variable]

Communication layer packet:

Application ID [2b]	Message format [1b]	Hop count [1b]	Security layer packet [variable]	Checksum of packet [8b]