Efficient Query Distribution and Positioning in Public Sensing Systems

Von der Fakultät Informatik, Elektrotechnik und
Informationstechnik der Universität Stuttgart
zur Erlangung der Würde eines Doktors der
Naturwissenschaften (Dr. rer. nat.)
genehmigte Abhandlung

vorgelegt von

Patrick Baier
aus Buchen (Odenwald)

Hauptberichter: Prof. Dr. Kurt Rothermel

Mitberichter: Prof. Dr. Friedemann Mattern

Tag der mündlichen Prüfung: 15.09.2015

Institut für Parallele und Verteilte Systeme (IPVS)
der Universität Stuttgart

2015

Acknowledgments

I would like to express my special appreciation and thanks to my advisor Professor Dr. Kurt Rothermel, who has made this work possible in his group. I would like to thank him for encouraging my research and for allowing me to grow as a research scientist. I would also like to thank Professor Dr. Friedemann Mattern for taking the time to review my dissertation.

Furthermore, I would like to thank my colleagues from the distributed systems group, which inspired me with new ideas and were always open to provide insightful feedback. Many thanks to Frank Dürr, Harald Weinschrott, Damian Philipp, Marius Wernke, Naresh Nayak and Stefan Föll.

I also would like to thank the German Science Foundation, that promoted my research under grants *FR 823/25-1* and *RO 1086/17-1* and provided me the chance to present my results to the research community on various international conferences.

A special thanks goes to my family. Thank you for your continuous support and encouragement over all these years.

Contents

List of Abbreviations

DR	Dead reckoning
GPS	Global positioning system
IPS	Inertial positioning system
LBA	Location-based application
LS	Location server
MDP	Markov decision process
ONE	Opportunistic network simulator
pdf	Probability density function
PS	Public sensing
RSSI	Received signal strength indication

Abstract

Within the last few years mobile phones have evolved from traditional communication devices to powerful computational platforms that are equipped with a variety of sensors. For instance, a modern phone comes with a barometer, a light sensor, a magnetic field sensor and a microphone. These sensors turn mobile phones into powerful networked sensor platforms that can be used for capturing sensor data. The captured data can be aggregated and analyzed to monitor environmental phenomena such as noise pollution, for instance. Using mobile phones for sensing is termed in the literature as *Public Sensing* (PS) and has recently attracted increasing interest in the research community.

One big challenge of PS is the high energy consumption that it imposes on mobile devices. Running a PS system can quickly drain the battery of a device, which is a major concern that prevents people from participating in PS. To tackle this issue, this thesis provides energy efficiency algorithms for two of the most energy-intensive operations in PS. First, it introduces a novel *query distribution approach* that significantly reduces the communication energy of a device when receiving queries for sensor data from the infrastructure. This approach limits the set of devices that receive a sensing query, in contrast to existing approaches, which distribute queries to all available devices. As part of this contribution, a novel position update protocol is presented that increases the efficiency of existing protocols by sending updates opportunistically together with other messages. Secondly, this thesis presents an *efficient position sensing approach* that controls the positioning system of a mobile device such that the energy for positioning is minimized. While existing PS systems assume that a device's positioning system is always-on, this approach temporarily disables the positioning system of a device without reducing its sensing effectiveness.

Deutsche Zusammenfassung

Innerhalb der letzten Jahre haben sich herkömmliche Mobiltelefone zu leistungsstarken Kleincomputern weiterentwickelt, die mit einer Vielzahl an Sensoren ausgestattet sind. Ein modernes Smartphone verfügt beispielsweise über einen Barometer, einen Helligkeits- und Beschleunigungssensor, einen Kompass, ein Mikrofon sowie ein Positionierungssystem. Darüber hinaus ermöglichen die zunehmend steigenden Datenübertragungsraten in Mobilfunknetzwerken eine schnelle und dauerhafte Datenverbindung in die Infrastruktur.

Diese technischen Voraussetzungen ermöglichen es, Mobilgeräte als mobile Sensorknoten zu nutzen, welche Umgebungsdaten aufzeichnen und in die Infrastruktur übertragen. Beispielsweise können Audiosignale von mehreren geografisch verteilten Smartphones aufgezeichnet und zentral analysiert werden, um Lärmquellen in einer Stadt zu identifizieren. Die Idee bereits vorhandene Mobilgeräte als mobile Sensorknoten zu nutzen wird in der Literatur als Public Sensing (PS) bezeichnet und ist in den letzten Jahren auf wachsendes Interesse in der Wissenschaftsgemeinde gestoßen. So ermöglicht PS das Aufzeichnen von Sensordaten ohne vorherige Investitionen und stellt daher eine interessante Alternative zu klassischen Sensornetzwerken dar, speziell in dicht besiedelten Gebieten wie in Stadtzentren. Um ein PS-System zu realisieren müssen zuvor eine Reihe von Herausforderungen gelöst werden, welche über das Forschungsgebiet der klassischen Sensornetzwerke hinausgehen. Besonders kritisch ist hierbei die Benutzerakzeptanz. Um Sensordaten für das PS-System bereitzustellen, muss der Besitzer eines Mobilgeräts seine knappen Geräteressourcen zur Verfügung stellen, ohne dafür einen unmittelbaren Nutzen zu erhalten. So verbraucht ein Mobilgerät in einem PS-System Energie für das Aufzeichnen von Sensordaten, für die Positionierung und für die Kommunikation mit der Infrastruktur. Meh-

rere Studien belegen, dass die dauerhafte Ausführung dieser Operationen die Akkulaufzeit eines Mobilgeräts enorm verkürzt. Folglich sind Algorithmen zur energieschonenden Ausführung dieser Operationen eine der Hauptansatzpunkte um die Benutzerakzeptanz von PS-Systemen zu erhöhen.

Während sich bisherige Forschungsarbeiten weitestgehend auf Ansätze zur Effizienzsteigerung beim Aufzeichnen der Sensordaten beschränken, wurde die Optimierung der Datenübertragung und der Positionierung in PS-Systemen bisher nicht umfassend untersucht. Existierende Ansätze nehmen beispielsweise an, dass der in der Infrastruktur operierende PS-Server mit den Mobilgeräten mit Hilfe eines Broadcast-Ansatzes kommuniziert. So wird eine Anfrage an das PS-System, die Sensordaten von einem bestimmten Ort anfordert, immer an alle Mobilgeräte gesendet, ohne die geografische Verteilung der Geräte zu berücksichtigen. Dieser Ansatz skaliert jedoch nicht in PS-Systemen, die aus mehreren Millionen Mobilgeräten bestehen, da jede Kommunikation mit der Infrastruktur Energie auf den Geräten verbraucht. Des Weiteren nehmen existierende PS-Systeme an, dass Mobilgeräte zu jeder Zeit ihre aktuelle Position kennen, um die aufgezeichneten Sensordaten zu georeferenzieren. Mehrere Untersuchungen stimmen jedoch darüber ein, dass eine dauerhafte Positionierung die Akkulaufzeit eines Mobilgeräts auf wenige Stunden reduzieren kann.

Diese Dissertation erweitert den Stand der Wissenschaft in der PS-Forschung, indem sie energieeffiziente Algorithmen für die folgenden offenen Probleme vorstellt: (1) Einen Ansatz zur effizienten Verteilung von Sensordatenanfragen an Mobilgeräte. Dieser Ansatz identifiziert für jede Anfrage eine begrenzte Teilmenge an Mobilgeräten, welche diese Anfrage erhalten. Im Vergleich zu existierenden Verfahren kann somit bis zu 50 % der Kommunikationsenergie auf Mobilgeräten eingespart werden. Zudem wird im Rahmen dieses Ansatzes ein effizientes Positionsupdateprotokoll entwickelt, welches die Energieeffizienz bestehender Verfahren um bis zu 70 % steigert. (2) Einen adaptiven Positionierungsalgorithmus, welcher die Positionierung eines Mobilgeräts so optimiert, dass die Positionierungsenergie um bis zu 90 % reduziert wird, ohne jedoch die Effektivität des PS-System hinsichtlich der Menge der gesammelten Sensordaten zu reduzieren.

1 Introduction

1.1 Motivation

Within the last few years mobile phones have evolved from traditional communication devices to powerful computational platforms that are equipped with a variety of sensors. For instance, a modern phone comes with a positioning sensor, a barometer, a light sensor, a magnetic field sensor and a microphone. Moreover, fast communication technologies like 3G/4G networks together with cheap flat rates are available, which has led to an "always-on" usage pattern where mobile phones are constantly connected to the Internet. These trends effectively turn mobile phones into powerful networked sensor platforms in which each device can act as a sensor node that reads sensor data. The captured data can be aggregated and analyzed to monitor some environmental phenomenon, just like in traditional sensor networks.

Using mobile phones for sensing is referred in literature as "Public Sensing" (PS) [BWD+11; PDR11; BDR12a] and has attracted an increasing interest in the research community within the last few years. The fact that PS can be used on-demand to gather sensor data from large geographical areas without any big upfront investments makes it an attractive alternative to classical sensor networks. This is especially true in densely populated areas such as urban centers, where typically a large number of mobile devices are available. The general vision of PS is to be able to answer ad-hoc queries for sensor data – e.g., "what is the current noise level in front of the library" – in a timely, efficient and accurate manner. To realize this vision, a number of challenges have to be tackled that are beyond the scope of traditional sensor network research. Obviously, the most critical requirement for a wide scale deployment of PS systems is user acceptance. In order to contribute sensor data, a mobile device

owner has to spend his scarce device energy without getting a direct benefit in return. More precisely, a PS system typically requires a device not only to read sensor data but also to run a positioning system to geo-reference the captured data. Additionally, a device needs to communicate with the infrastructure in order to receive queries for sensor data and to upload sensor data. Several studies [BBV09; KLG+09] agree that these operations are very energy-intensive and can significantly shorten a mobile device's battery lifetime. Hence, a major prerequisite for reaching user acceptance is that a PS system performs these operations in an energy-efficient manner. If this prerequisite is not fulfilled, a PS system lacks user support. As a result, the system has only very limited geographical coverage.

So far, first approaches exist that optimize the energy efficiency of PS systems by decreasing the energy for reading sensor data. For this purpose, they provide algorithms to coordinate sensing operations [WDR10; PDR11; STZ12], which try to avoid redundant sensor readings. While these approaches focus on sensing efficiency, the optimization of two other energy consuming operations have been neglected in existing PS systems, namely *query distribution* and *device positioning*. On the one hand, existing solutions assume that the PS server, which coordinates the sensing process, communicates with the devices in a broadcast fashion. As a result, sensing queries that contain sensing instructions for mobile devices are sent to all devices, not taking into account if a device is actually well suited to collect data for a given query. Considering that a PS system may contain several million devices, this approach does not scale and generates a big energy overhead on devices when receiving queries. On the other hand, existing solutions assume that mobile phones continuously fix their position, e.g., with the Global Positioning System (GPS), to geo-reference the sensed data. However, it has been shown that this approach drains a device's battery very quickly [KLG+09].

This thesis advances the current state of the art in PS research by providing energy efficiency algorithms for query distribution and device positioning. Hence, the contribution of this thesis is twofold: First, it provides an efficient

query distribution algorithm that identifies for a given sensing query a limited target set of mobile devices that should receive this query. One part of this approach is a novel position update protocol that increases the efficiency of existing protocols by sending updates opportunistically together with other messages.

Secondly, this thesis introduces an efficient position sensing algorithm that controls the positioning system of mobile devices such that positioning energy is reduced significantly while not decreasing the effectiveness of sensing.

1.2 Background on Public Sensing

To get a thorough understanding of the technical contributions of this thesis, which are elaborated in the next section in detail, it is necessary to be familiar with the general idea of PS and the research that has been done in that area. For this purpose, Section 1.2.1 gives a short overview of PS in general, before Section 1.2.2 introduces some concrete examples of PS systems that have been already deployed. Since the contributions of this thesis are agnostic of any specific sensing application, Section 1.2.3 introduces the vision of a *generic* PS system, which follows the idea to have a single PS system that can be used to collect any kind of sensor data. Subsequently, Section 1.2.4 introduces the opportunities and the challenges that arise with PS, which lead to the central research goal of the thesis, which is energy efficiency in PS. To this end, Section 1.2.5 gives an overview of energy consuming operations in PS and how they were addressed by researchers so far.

1.2.1 Overview

The basic idea of PS is to opportunistically collect sensor data from mobile devices that are carried around by people on their daily routines. Considering the fact that nowadays almost every person carries a mobile device, PS enables a cheap and fast way to acquire environmental data from a large geographical area without requiring any deployment and maintenance of sensor nodes. The enabler for this new paradigm is the technical improvement in the hardware of

mobile phones, which are nowadays equipped with a large variety of sensors that can be used for data collection. For instance, Apple's iPhone 6 is equipped with an ambient light sensor, a proximity sensor, dual cameras, a GPS sensor, an accelerometer, a microphone, a barometer and a gyroscope [App14]. Additionally, external sensors can be attached that further increase the number of available sensors on a device [BSC+12], while new devices like tablets or wearables (e.g., smart watches or glasses) enjoy increasing popularity and also come with integrated sensors that can be used in PS systems.[1] Hence, the number of devices that can be used to collect data for PS continues to increase.

The opportunities and challenges that arise from PS were first characterized by Abdelzaher et al. in their visionary paper "Mobiscopes for Human Spaces" [AAB+07] in 2007. Since then, a large number of research papers have been published around this idea, which has been termed "Public Sensing" [BWD+11; PDR11; BDR12a], "Public Urban Sensing" [CHK08] or "Mobile Phone Sensing" [LML+10]. Soon after identifying the opportunities and challenges of PS [CHK08; CEL+08], first real-world prototypes began to show its practical feasibility [ELK+08; TRL+09]. Within only a few years, a large number of different PS applications were deployed (see the next section for an overview).

The general idea of using mobile devices to capture sensor data comes in two different flavors which are *opportunistic* sensing [CEL+06] and *participatory* sensing [BEH+06]. The opportunistic approach is based on the idea to collect sensor data in the background without requiring any interaction by the device owner. For instance, a microphone sample is automatically recorded by a device when it passes a certain location without the device owner even noticing that data was captured. In contrast, the participatory approach prompts the user for interaction during the data collection. For instance, the PS system could ask the user to take a picture when reaching a certain place. This thesis only considers opportunistic sensing. Hence, when using the term PS in the following, we consider systems that do not require any interaction from the user.

[1] For simplicity, all devices that are suitable for PS will be referred for the rest of this thesis by the general term *mobile devices*.

1.2.2 Applications of Public Sensing

In general, the concept of PS can be applied in many different areas. To get an idea of these application domains, the following list gives an overview of PS systems that were presented in the literature so far. However, this is not a complete list of all deployed PS systems since the number of such systems is large and is still increasing. A more detailed overview of existing PS systems is given by Lane et al. [LML+10] or by Khan et al. [KXA+13].

Environmental Monitoring. The PS systems in this category use sensors of mobile devices to observe and analyze environmental phenomena. One of the most prominent examples in this area is the idea of using PS to create noise maps [RCK+10; DSJ13]. For instance, the EarPhone system [RCK+10] collects noise pollution readings from mobile devices to construct a noise map for a predefined urban area. While these approaches utilize the built-in microphone on a mobile device, other phenomena can only be observed by attaching external sensors to the device. The GasMobile system [HSS+12] creates ozone pollution maps by using external ozone sensors that are attached to HTC Hero devices. The Sundroid system [FKS+11] measures UV radiation with a dedicated sensor unit that is connected to a smartphone. The collected data can be aggregated for research studies or to warn the user if a critical amount of radiation for his body has been reached.

Apart from using mobile phones for sensing, other approaches use dedicated mobile sensing platforms for data reading. For instance, MAQS [JLT+11] uses a hardware system that includes a temperature, CO_2, humidity and a light sensor. This system is carried around by people in order to collect indoor environmental quality data. In contrast, Hasenfratz et al. [HSW+14] use mobile sensor nodes that are installed on top of public transport vehicles in order to collect data about ultrafine particles in the air.

Traffic Analysis. One other domain in which many PS systems have already been successfully deployed is the area of traffic analysis. The approaches in this field can be summarized as follows: Mobile devices that are inside vehicles

record trace data with the help of a positioning system such as GPS. This data is uploaded to a central server where it is analyzed and information about the current traffic state is inferred.

For instance, the VTrack system [TRL+09] estimates the travel time of a vehicle by aggregating position data from multiple devices. To calculate the travel time, it tries to estimate which road segments on the future route are delay-prone and accordingly adjusts the estimated travel time. Quite similar, Zhou et al. [ZZL12] propose a system that predicts bus arrival times. Rather than using dedicated hardware, they only utilize the bus passengers' mobile phones for analyzing bus travel times, which turned out to result in very accurate estimations for the arrival times. Instead of only observing traffic, other systems use the built-in sensors of a mobile device to monitor road surface conditions. The Nericell system [MPR08], for instance, uses the sensors of a mobile phone to detect potholes, bumps, braking and honking while driving. The Pothole Patrol [EGH+08] uses deployed hardware in cars that form a mobile sensing system. This system detects potholes and uploads the data to a central server, where the found data is clustered in order to reliably identify potholes. In contrast, the GasStation approach [ZWZ+13] tackles a quite different challenge. Using reported trajectories from GPS-equipped taxicabs, the approach detects visits to gas stations and uses this data to predict waiting times at gas stations.

User-centric Sensing. Apart from PS systems that observe phenomena over a large geographical area, there is a large number of sensing approaches that only monitor data that is concerned with the mobile device's owner. For instance, the StressSense system [LFR+12] recognizes stress from human voices with the help of microphones that are embedded in smartphones. Quite similar, the EmotionSense system [RMM+10] uses mobile phone sensors to sense individual emotions and activities. The goal of this platform is to easily support social and psychological experiments. Other systems in this area provide sleep monitoring [CLC+13] by analyzing smartphone usage and sensors, or they assists elderly people by implementing a fall detection system [DBY+10] that is based on mobile phone sensor data.

Indoor Mapping. While location-based applications like restaurant finders are already popular in outdoor scenarios, one problem that prevents the deployment of such applications for indoor scenarios is the lack of publicly available indoor floor plans. In general, the creation of indoor floor plans is a manual and labor intensive task. Therefore, several PS-based systems were proposed that automatically derive indoor floor plans from pedestrian traces that were captured and uploaded by mobile devices. These approaches either differ by the type of positioning system that is used on the device [SCC12; JXP+13] or by the floor plan derivation algorithm that is used for creating the map [AR12; PBD+14]. A concrete example of such system is given in Section 1.4, when we have a closer look at the ComNSense project, in which the work of this thesis is embedded.

1.2.3 Generic Public Sensing Systems

The systems introduced in Section 1.2.2 all have one thing in common: They were built for only one specific sensing application. For instance, the noise mapping system provides algorithms to extract audio signals from mobile devices but is not applicable for air quality monitoring. Hence, if a mobile device should take part in more than one sensing campaign, this would require that it runs an individual sensing application for each sensing system. Furthermore, each sensing application would communicate with its individual PS server in the infrastructure. Given the large variety of PS applications, this is obviously not a scalable solution. Moreover, device users are in general rather reluctant to host a large number of different PS applications on their devices.

To overcome this issue, several approaches [WDR09; TKT+10; ABC+13; CFB+13] propose to have only one PS system that comes with a generic query interface. In such a system, only one sensing application runs on the device and is responsible for reading all kind of different sensor data. Sensor data applications like air quality monitoring can then be developed on top of this system and can query the generic PS system for the required sensor data. The architecture of such a system is as follows (see Figure 1.1): A server in the infrastructure takes sensing queries from sensor data applications, which it then distributes to

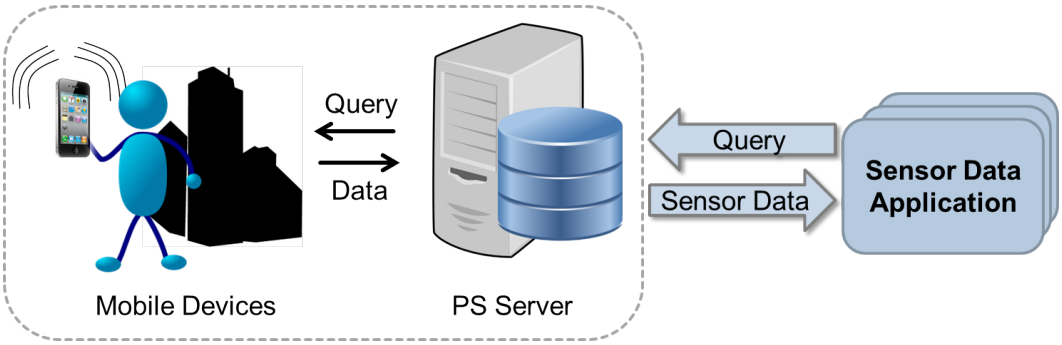


Figure 1.1: Architecture of a generic Public Sensing system.

mobile devices. If a device has successfully captured the queried data, it sends the sensor data back to the PS server where it is returned to the sensor data application.

The concepts that are introduced in this thesis follow the idea of such a generic PS system. This has the advantage that they can be applied in a large number of sensing scenarios and not just in a specific system. Hence, we will have a more detailed look on the concepts of such a system in the system overview that is given in Chapter 2.

1.2.4 Opportunities and Challenges

Soon after the idea of collecting sensor data with mobile devices was proposed, researchers started to realize that this new way of acquiring sensor data not only brings opportunities but also raises many challenges. In the following, we will first analyze the advantages of PS over traditional sensor networks and then have an overview of the challenges that PS imposes.

The traditional way of collecting environmental data is the deployment of a sensor network consisting of sensor nodes. Although there is a large body of research on this topic (see [ASS+02] and [RM04] for an overview), there are still several challenges that need to be tackled: Initially, sensor nodes have to be deployed, which requires substantial manual effort. Furthermore, sensor nodes have to be maintained. For instance, sensor nodes are typically powered by

batteries, which have to be replaced manually. Finally, the sensor nodes' limited communication range imposes geographical constraints on the deployment of the nodes, since data is typically communicated to a sink node via ad-hoc forwarding between the nodes [AK04].

In comparison, PS inherently overcomes many of these limitations. Leveraging existing mobile devices avoids the deployment of sensor nodes, since mobile devices are already available in the general public, especially in densely populated areas such as urban centers. Hence, no big upfront investments are necessary to deploy a PS system as devices are maintained by their owners. Furthermore, a mobile device typically comes with a fast communication uplink, which enables the upload of sensor data from almost everywhere.

The aforementioned characteristics of PS show that the deployment of a large-scale PS system is a very promising solution to foster the collection of environmental data. However, so far no PS system is available that has enough participating devices to reach high geographical coverage. This can be attributed to the fact that there are still many open problems that discourage people to run a PS application on their devices. The following list details three of the major concerns that are still open research questions:

- The *privacy* aspect is inherent in PS systems. Uploading geo-referenced sensor data to a PS server also means that sensitive location information of a user is revealed. First solutions to tackle this problem are already available [CRK+11; DS13; AK14; GGP14], while research on finding an easy to deploy solution is still ongoing work.

- Giving up privacy is usually a restriction that people do not accept, unless they get something in return [KH08]. However, in PS systems a user typically does not receive a direct benefit for contributing the resources of his device to the data collection process. Hence, *incentive mechanisms* are needed in order to convince people to participate. For instance, the collected sensor data can be made accessible to the people who collected the data. Projects like OpenStreetMap [OSM] show that many people are

willing to voluntarily contribute data to support a public project. However, the number of such contributors is small and not sufficient to run a large-scale PS system. Finding a more general incentive mechanism for PS is still ongoing work, though a number of papers have already proposed first approaches towards reaching this goal [YXF+12; LC13; AKS+13; TTO+14].

- Another open problem in PS is *energy efficiency*. The price for the versatility of modern mobile devices is their high energy demand, i.e., a typical smartphone runs out of battery within a few days. Very energy-intensive applications can exhaust the battery even within a few hours. As result, a user is typically aware of energy-draining applications and naturally tries to avoid them. Energy efficiency is therefore a prerequisite for the deployment of PS systems, otherwise device owners are not willing to participate.

This thesis focuses on the last of these research questions by providing algorithms and concepts to increase energy efficiency in PS systems. To have a deeper understanding of the different aspects that are related to energy efficiency, we elaborately look in the following section on different approaches for achieving energy efficiency in PS.

1.2.5 Energy Efficiency

To increase energy efficiency in PS systems, the research community is focusing on the design of energy-efficient algorithms for all tasks a mobile device has to execute in a PS system. The following list addresses each of these tasks in detail and shortly describes the solutions that were proposed so far.

Data Sensing. One central task that a mobile device has to execute in PS is the reading of sensor data. Depending on the sensor that the device uses for reading sensor data, this can be very energy consuming. So far, several approaches have been proposed to reduce the energy spent for data sensing. To begin with, Lane et al. propose to delay data sensing until the device wakes up from sleep mode triggered by some other process [LCZ+13]. This reduces energy consumption since a device does not need to wake up from sleep mode to read sensor data.

Another set of approaches enables energy savings by completely avoiding the sensing operation. This can be achieved by offloading the sensing to fixed sensors that are embedded in the environment [REL+14], or by coordinating the mobile devices to avoid sensing redundant data [WDR10; PDR11; STZ12]. While all of these approaches do not require any changes on a device's hardware, it has been shown that the use of a dedicated low-power processor is another way to significantly increase energy efficiency of sensing [PLL10; RPK+12].

Data Uploading. After successfully reading sensor data, a mobile device has to make the data available by uploading it to the PS server that is deployed in the infrastructure. The size of this data can range from a single temperature value to a high resolution image. Hence, energy-efficient sensor data upload strategies are an important aspect to increase overall energy efficiency. For uploading big amounts of sensor data, it has been shown that waiting for the availability of a WiFi connection can be much more energy-efficient than uploading the data immediately via the 3G network [LHZ+13]. Furthermore, the upload of sensor data can be fostered by providing rewards to people, which need to sacrifice their device energy for uploading [CHH14]. Other research tackles the question, when sensor data should be uploaded to let the PS server track the changes of a certain phenomena (e.g., only upload sensor data if the local temperature changes). For instance, energy efficiency can be increased by scheduling the sending of update messages for sensor data with distance-based or prediction-based update algorithms [LR01; MPF+10; FHR12].

Query Distribution. To know where to read sensor data and which sensor should be used for reading the data, a mobile device needs to be aware of the sensing queries that are currently present in the PS system. As we have seen in Figure 1.1, such a query is issued from a sensor data application to the PS server, which then distributes it to the mobile devices. The receiving of a query forces a device to power up its communication interface. However, it has been shown that receiving data consumes a significant amount of energy on a device, even if the payload of the data packet is very small [BBV09].

Most existing PS systems perform query distribution by simply broadcasting

a sensing query to all devices that participate in the PS system. Obviously, this simple strategy wastes energy for devices that are not needed for sensing, for instance, due to their geographic distance to the sensing range of the query. This is especially in large-scale PS systems a problem, which may receive several sensing queries per second. In such a system, the battery of a mobile device would quickly drain because the device is kept busy receiving sensing queries.

First solutions try to tackle this problem [RES10; RNL11; CFB+13] by analyzing historic information about a user's movement pattern in order to extract places were users regularly reside. Knowing where users spent most of their time can help to infer which devices are likely to capture data for a sensor query that requires data from a certain location. However, these approaches require that a user agrees to store a privacy critical user profile of its movement history. Furthermore, they do not consider the energy that is needed on the devices to generate these profiles and to keep them up-to-date on the PS server. To overcome these drawbacks, one contribution of this thesis is an approach for the efficient distribution of sensing queries. This approach works without the need for analyzing regularly visited places and includes a dedicated position update protocol. This protocol is designed such that the energy for providing position information from devices to the PS server is minimized. Rather than going into detail about this contribution at this place, we have a detailed look on it in the next section.

Device Positioning. Mobile devices require a positioning system to georeference sensor data and to check if sensor data is queried at a device's current position. Existing PS systems assume that a device constantly compares its current position with the locations from which sensor data is queried. If the device comes close enough to one of these locations, it reads data for that query. These approaches require continuous positioning, which is a energy consuming operation [KLG+09] that can exhaust a device's battery within a few hours. Hence, device positioning is an open problem which will be tackled by this thesis, as we will see next.

1.3 Contributions

The main contribution of this thesis is to provide energy-efficient optimizations for two of the above identified open problems, namely *query distribution* and *device positioning*. As we will see later on, one prerequisite to implement an algorithm for efficient query distribution is to know the spatial distribution of the mobile devices. For this purpose, a device can run a position update protocol that periodically sends position updates to the PS server. However, existing position update protocols consume too much energy for sending these updates. To tackle this issue, one further contribution of this thesis is an *opportunistic position update protocol* that increases the energy efficiency of prevailing position update protocols.

In the following, all contributions of this thesis are described in more detail.

1. Opportunistic Position Update Protocol. This thesis advances the state of the art in position update protocols by introducing an opportunistic extension to existing update protocols, which reduces the energy consumption of these protocols significantly. This approach is motivated by the fact that the cellular network interface of a mobile device has a non-linear power characteristic, i.e., the power states of the mobile network interface have certain transition delays that impact the energy consumption of individual messages. For instance, sending two messages directly after each other is more efficient than sending the second message one minute after the first message. Existing update protocols are optimized under the assumption that the energy consumption for different position update messages is equivalent. In contrast, the presented approach takes the energy characteristics of the mobile network interface into account which allows for an energy-aware scheduling of position updates. This thesis shows that the energy consumption on the mobile device can be reduced significantly using an opportunistic update strategy, which sends position updates together with messages of other applications. In order to develop this protocol, the following contributions are made: (1) An empirical study is conducted that shows that communication patterns of mobile devices are in general well-suited for an

opportunistic update strategy. (2) A Markov decision process is presented that determines the optimal time to schedule the transmission of position updates. (3) The evaluation results show that this opportunistic protocol improves the energy efficiency by up to 70 % in comparison to existing update protocols. Main parts of this contribution have been published before in [BDR13b].

2. Efficient Query Distribution. This thesis introduces an efficient sensing query distribution algorithm that forwards a given sensing query to only those devices that are likely to actually sense data for that query. This reduces the overall energy consumption of the PS system significantly, since the communication overhead is reduced for all devices that do not receive the query. In contrast, existing approaches broadcast sensing queries to all devices in the system and, hence, impose energy overhead on every device. For instance, consider a query for sensing the temperature and noise level at the Big Ben in London within the next ten minutes. Broadcasting this query to all mobile devices in London certainly involves many devices which cannot reach Big Ben within the given deadline and, hence, waste energy resources on these devices.

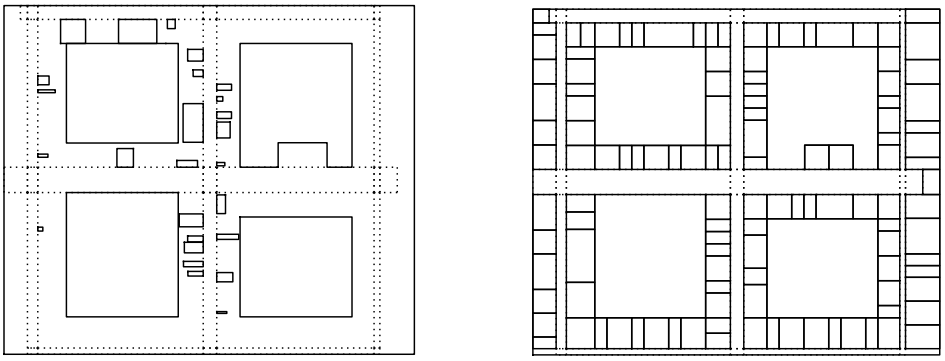
To avoid this waste of energy, the proposed approach utilizes the information that is provided by the opportunistic position update protocol. Having access to location information of the mobile devices, the PS server can choose a subset of devices that are well-suited for receiving a given sensing query. However, this selection is non-trivial because the server does not know the future movement paths of the devices. Hence, the server cannot be perfectly sure that a device actually comes close enough to sense data for a query. To tackle this issue, the query distribution algorithm utilizes a probabilistic sensing model. This model estimates the sensing success of each device by taking into account all possible movement paths of a device. Based on this model, the presented algorithm selects a minimum set of devices that should receive a given query such that the chances of sensing data for this query are not decreased. The evaluations show that this optimization reduces the energy consumption on mobile devices by up to 50 % compared to an implementation that broadcasts a given sensing query to all available mobile devices. Main parts of this contribution have been

published before in [BDR13a].

3. Efficient Outdoor Position Sensing. In existing PS systems, devices continuously track their position to know when to start sensing. Hence, a device has to continuously query its GPS sensor when moving outdoors, which is not acceptable in terms of energy consumption. This thesis proposes an efficient positioning approach that significantly reduces positioning energy by only selectively turning on the GPS sensor. For this purpose, it calculates the points in time when the GPS sensor should be queried for a new position. The challenge is to preserve the sensing effectiveness of the PS system since the reduced number of position fixes might result in less data readings. The evaluation of the approach shows that it reduces GPS positioning energy by up to 90 % without any reduction in sensing effectiveness. Main parts of this contribution have been published before in [BWD11], [BWD+11] and [BDR12a].

4. Efficient Indoor Position Sensing. For indoor positioning, devices typically rely on an inertial positioning system (IPS), as GPS is indoors not available and infrastructure-based positioning methods are deployed only very scarcely. However, running an IPS is highly energy consuming and drains a mobile device battery quickly. In contrast to a GPS sensor, an IPS cannot be easily temporarily disabled, since it requires an absolute position every time it is started. To increase the energy efficiency of positioning also indoors, this thesis proposes an approach that overcomes this restriction. More precisely, it introduces an indoor repositioning algorithm that allows to temporarily turn off the IPS. This is enabled by so-called WiFi anchor points that help to reactivate the IPS by providing new initial positions. The evaluation shows that this approach reduces the energy for indoor positioning by up to 25 % compared to a naive positioning system that only relies on continuous IPS positioning. Main parts of this contribution have been published before in [BPD+14].

Note that for all above listed publications, the idea, the concept and the implementation were developed by the author of this thesis. Moreover, large parts of the papers were written by the author, while Frank Dürr and Kurt Rothermel contributed to the refinement and presentation of the concepts.



(a) Trace-based (b) Grammar-based

Figure 1.2: Derivation of an indoor floor plan from indoor pedestrian traces.

1.4 The ComNSense Project

Most of the contributions of this thesis where motivated by the *ComNSense*[2] research project, which is funded by the German Research Foundation (DFG). To give some background information on this project, this sections introduces the project in more detail.

The goal of ComNSense is to develop a universal method for the automated generation of interior models with the help of mobile devices. The basic idea is to opportunistically collect indoor traces from devices that are carried and owned by the general public to automatically derive indoor floor plans. Since these traces are in general noisy and drift-based, an indoor grammar for building interiors is used to increase robustness of the data acquisition and to improve the resulting floor plan in terms of completeness and accuracy. The MapGENIE approach [PBD+14] shows how this idea enhances the indoor mapping process significantly, i.e., produces detailed indoor floor plans from only a small set of traces. Figure 1.2 shows the derivation of an indoor floor plan in two different versions: Only relying on information that is provided by trace data (Figure 1.2a) and using the information that is stored in the indoor grammar to enhance the floor plan generation (Figure 1.2b). The figure shows that the

[2] `http://www.comnsense.de`

indoor grammar obviously enhances the generated floor plan significantly. In numbers, the grammar-based approach identifies up to four times more rooms in a building than the purely trace-based approach, while at the same time it achieves a consistently lower error in the size of detected rooms.

Another goal of ComNSense is the energy-efficient collection of the indoor traces that are needed for deriving the indoor floor plan. As in a general PS approach, ComNSense relies on the acceptance of the device owners, who are otherwise unwilling to contribute indoor traces to the system. Based on this motivation, significant parts of this thesis were developed. However, the contributions of this thesis are not specific to indoor mapping. Only the proposed indoor positioning approach is evaluated with the help of the ComNSense system. All other contributions were developed independently of the indoor mapping scenario.

1.5 Structure of the Thesis

The remainder of the thesis is as follows: Chapter 2 gives a detailed description of the system architecture upon which the technical parts of this thesis built. This also includes a system model, which defines the assumptions that are made about each component of the system. Chapter 3 contains the definition of a basic sensing system, which serves as reference system for evaluating the efficiency of the contributions. Chapter 4 introduces an opportunistic position update protocol, which is a prerequisite for the efficient query distribution that is subsequently presented in Chapter 5. Chapter 6 introduces an efficient position sensing approach for outdoor scenarios, before Chapter 7 shows how this approach can also be applied in indoor scenarios. Finally, Chapter 8 concludes this thesis and gives an outlook on future work.

2 System Overview

The goal of this chapter is to give an overview of the system model and the architecture of the PS system. It starts with the presentation of the system model in Section 2.1 and the definition of the query semantics in Section 2.2. Subsequently, Section 2.3 outlines the architecture of the PS system, which includes all components that are presented in this thesis and their relations with each other.

2.1 System Model

To define the system model on which the contributions of this thesis rely, this section includes all assumptions that are valid throughout the rest of this thesis.

The system model includes three components:

1. A set of *mobile devices*, which read sensor data.
2. The *PS service* that coordinates the sensing process and communicates with the devices.
3. A *mobile communication network*, which allows the devices and the PS service to communicate with each other.

In the following, we have a detailed look on the assumptions that are made about each of these components.

2.1.1 Mobile Devices

Mobile devices are the key enabler for every PS system, since they collect sensor data and provide this data to the system. In general, a mobile device is carried

by its owner, who is moving with the device along the routes of his daily routine. Neither the speed nor the direction of this movement can be influenced by the system. In general, the movement of people is constrained by an underlying spatial structure. Depending on whether the device owner is outdoors or indoors, the movement path of a device is either restricted by a *road graph* or by an *indoor floor plan*. We assume that both are known by the PS system, as will be described later on in more detail.

A mobile device runs the PS application that executes sensor readings and communicates with the infrastructure. This application runs as a background process on the device and does not require any user interaction. For collecting sensor data, we assume that a device is equipped with a set of sensors, for instance, a microphone, a thermometer or a barometer. Each sensor can only read a certain type of sensor data.

For positioning, we differentiate between the cases when the device is moving outdoors or indoors. For outdoor positioning, we assume that a mobile device is equipped with a GPS sensor, which is currently the most accurate outdoor positioning system available in mobile devices. The position information that a device obtains from the GPS sensor is referred to as *position fix*. In general, a position fix is subject to a location error, which is around 8 m [Zan09]. This accuracy would be sufficient considering most PS scenarios. To simplify matters, the position error of GPS is neglected in the description of the technical concepts of this thesis. Nevertheless, all concepts in this thesis could be extended to take the GPS error into account by modeling a device position as a probabilistic uncertainty area, for instance with the help of a Gaussian distribution [LWG+09].

For indoor positioning, we assume that a device uses an inertial positioning system (IPS) to track its position indoors. Since inertial positioning in general suffers from drift errors that accumulate over time, we assume to availability of a foot-mounted inertial measurement unit that detects steps with the help of a Zero-Velocity-Update protocol [Fox05]. Furthermore, knowledge about the outline of the building exterior is available that can be used to align IPS traces [PBD+14]. Moreover, we require that a device has a WiFi interface to measure

the signal strength of WiFi signals.

2.1.2 PS Service

The PS service operates in the infrastructure and runs distributed over several PS servers that are connected via a wired network. Each PS server is responsible for the mobile devices in a certain geographical area, which is referred to as the server's *service area*. To get a better idea of such a service area, we could think of a PS server whose service area covers the inner city of Stuttgart, which roughly has an area of $15\,\mathrm{km}^2$. Since geographical load balancing is not subject of this work and has been investigated intensely in other works [XEC+07], we make no assumptions about the geographical partitioning of the service areas of different PS servers. To simplify the description of the concepts of this thesis, we follow the approach of other authors [WDR10; PSA+13] and describe the concepts in this thesis when only considering one PS server. Hence, we assume that all mobile devices move within the service area of this server and that all sensing queries are inside of this service area.

To communicate with the mobile devices in the service area, a PS server uses a mobile communication network, which is described next.

2.1.3 Mobile Communication Network

In order to exchange control and sensor data between the PS server and the mobile devices, a PS system requires a wireless communication network. However, it cannot be assumed that the entire service area is covered by accessible WiFi networks. Hence, to ensure high network coverage and availability, we assume that the server and devices communicate via a cellular mobile network like UMTS or LTE.

Sending and receiving data consumes energy on the mobile device. To characterize a device's energy consumption for mobile communication, we follow the empirical results given in [BBV09; LZZ11; DB12]. They show that the energy for transferring small pieces of data is negligibly small in comparison to the energy

overhead that is needed to power up and down the communication interface. The details of this energy model are not explained at this point, since they are introduced along with the position update protocol in Chapter 4.

2.2 Query Semantics

This section describes the query semantics that are valid for all the approaches presented in this thesis. A sensing query q is described by the parameters $q = \langle p, range, type, t_d, \sigma \rangle$. Throughout this thesis, we use a dot to separate a query and its parameters, e.g., $q.range$ denotes the *range* parameter of query q.

The semantics of the parameters are as follows:

- A query has a *sensing range* that describes the area from where sensor data should be read. This sensing range is defined by a geographical center point p and a radius *range* that defines a circular area that surrounds p. The geographical reference system to describe the coordinates of point p is not important for the concepts introduced in this thesis. Hence, we assume p to be given by absolute 2D coordinates with respect to a predefined origin.

- Each query is associated with a certain type of sensor data. This data can only be captured by a certain *sensor type*, referred to as *type*. For instance, this could be the microphone when a sound sample should be recorded or the thermometer for reading temperature data.

- To define the temporal requirements of a query, each query defines a *sensing deadline*, referred to as t_d. This value depicts the time until when the sensor data must have been captured. The value of t_d is given as an absolute point in time. After t_d the sensing query is considered as failed, if not at least one device captured data for the query.

- Optionally, a *quality parameter* $\sigma \in (0, 1]$ can be provided with a query. The issuer of the query can specify this value in order to define the amount of system resources the PS system can utilize to answer its query. In

a real deployment this value could be related to a payment system, for instance. The higher the value σ, the more likely it is that the PS system can successfully answer the query. We will see in Chapter 5 how parameter σ influences the semantics of the system. If this parameter is not specified, it is set to $\sigma = 1$ by default. This means that the PS system utilizes all available resources to answer the query.

A mobile device m can read data for a query q that it has received from the PS service at time t_q, if the two following conditions are fulfilled:

1. $\exists t \in [t_q, q.t_d] : d_{Eucl}(q.p, p_m(t)) \leq q.range$, where $d_{Eucl}(\cdot, \cdot)$ denotes the Euclidean distance and $p_m(t)$ the position of m at time t.

2. $q.type \in m.types$, where $m.types$ denotes the set of sensors that are available on m.

We define a query q to be *satisfied* as soon as there is *one* device that has read data for q. In contrast to other approaches [WDR09; WDR10] which require more than one sensor reading to satisfy a query, we explicitly limit our considerations to the case when one sensor reading is sufficient. Hence, we only capture one reading and leave it up to the issuer of the query to decide if the quality of the retained sensor reading is sufficiently high. If this is not the case, the system can be queried again to get more sensor data.

To simplify matters, we consider for the rest of this thesis only the case in which sensor data for a certain sensor type is queried. The set of all mobile devices in the PS system that are equipped with this sensor is denoted as M. Note that the algorithms that are presented in this thesis also work for the general case when various kinds of sensor data are queried. In this case, all concepts can be extended by defining individual sets of devices M_t for each sensor type t, on which the later on presented algorithms are then applied.

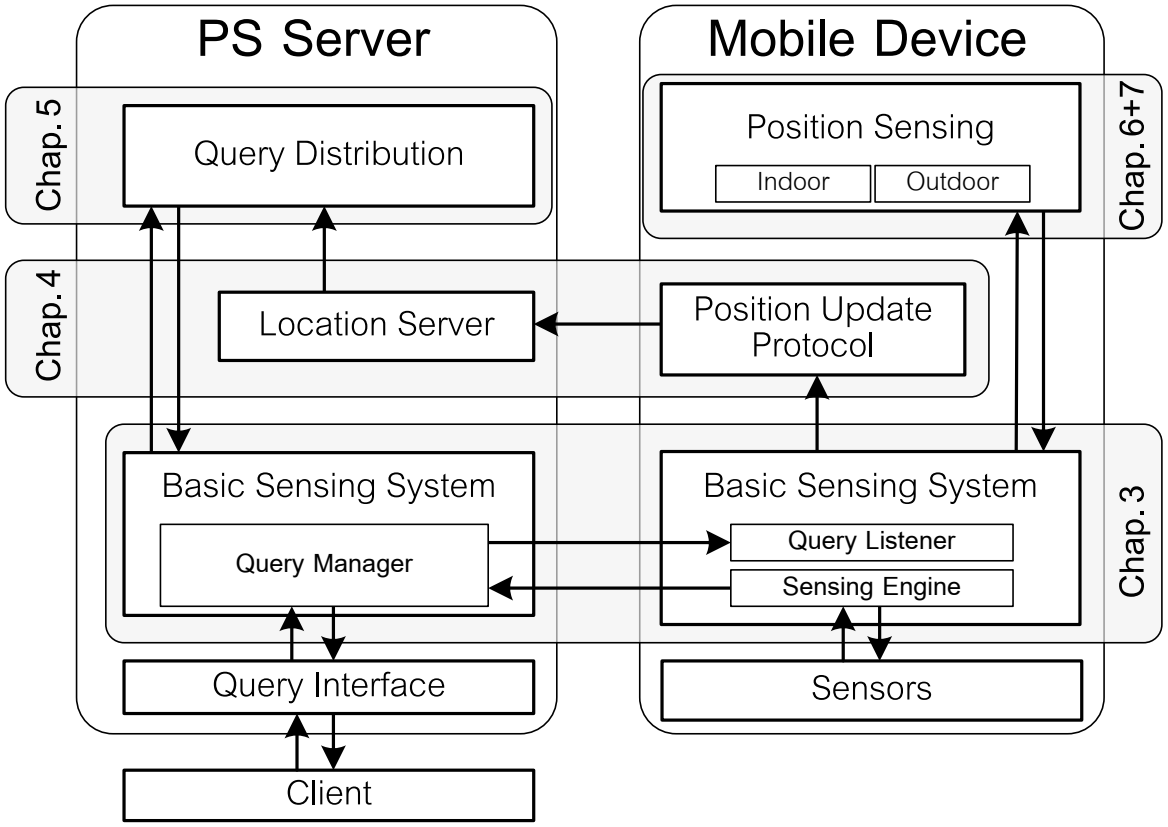


Figure 2.1: Overview of the architecture of the Public Sensing system.

2.3 System Architecture

This section gives an architectural overview of the components of the PS system that are presented in this thesis. Each component executes a certain task and provides an interface for interacting with other components. Figure 2.1 gives an overview of these components. Each component either runs on the mobile device or on the PS server in the infrastructure. The following sections briefly summarize the function of each component and its interaction with other components.

2.3.1 Basic Sensing System, Query Interface and Sensors

The *basic sensing system* consists of two subcomponents, one running on the PS server and the other running on the mobile device. On the server, it implements the *query manager*, which is responsible for coordinating the execution of sensing queries. On the mobile device, it consists of the *query listener* and the *sensing engine*, which are responsible for the interaction with the server and for reading sensor data, respectively. This basic sensing system is the common foundation of all the optimizations that are introduced in this thesis. Note that it also defines a very basic algorithm for query distribution and position sensing that are replaced by the optimized versions of these operations in the course of this thesis (see the following subsections 2.3.3 and 2.3.4).

The query manager of the basic sensing system is directly connected to the *query interface*, which receives sensing queries from *clients*. The query interface is publicly accessible, for instance, it could be deployed as a web service. A client can request sensor data for a certain location by sending a *sensing query* to the query interface.

A client is an external entity that queries the PS system for sensor data. For instance, a client can be a human user or an application that processes sensor data. A client query is forwarded by the query interface to the query manager that is responsible for managing the execution of the query. On a device, the sensing engine is responsible for reading sensor data by accessing the device's *sensors*.

2.3.2 Location Server and Update Protocol

The *position update protocol* runs on the device on top of the basic sensing system and sends position update messages to the *location server* that is deployed on the PS server. Position updates are sent to inform the PS server where the device is currently located. For deciding when a position update should be sent, the update protocol uses position information that the basic sensing system provides by querying the device's positioning sensor. On the server side, the

location server provides the position information that is needed by the efficient query distribution component that is introduced next.

2.3.3 Query Distribution

The *query distribution* component is responsible for identifying a subset of devices that should receive a given sensing query. This decision is based on the information about devices' positions that is provided by the location server. The component is invoked by the basic sensing system whenever a new sensing query needs to be distributed. It then returns the set of devices that should receive this query.

2.3.4 Position Sensing

The *position sensing* component schedules the operations for sensing the device's position in indoor and outdoor scenarios. In *outdoor scenarios*, it provides the basic systems with timeout intervals by which the next GPS position fix can be delayed. In *indoor scenarios*, it controls positioning by providing an approach that relies on WiFi anchor points to temporarily disable energy consuming IPS positioning.

3 Basic Sensing System

This chapter presents the details of the basic sensing system, which is a fully self-contained PS system that uses no optimizations considering sensing query distribution and position sensing. More precisely, it implements a simple query distribution on the server and continuous positioning on the devices. This system will be extended in the course of this thesis by integrating the optimization algorithms that are introduced in subsequent chapters. Since this system reflects the basic assumptions that are implemented in existing PS systems, it serves as reference system for evaluating the energy efficiency of the optimization algorithms that are presented in this thesis.

As mentioned earlier, the design of the basic sensing system follows a generic PS system that is not restricted to a specific application scenario and can therefore serve all kind of sensor queries. The generic PS systems that were envisioned before by other authors [WDR09; TKT+10; ABC+13; CFB+13] propose similar architectures and sensing models. Hence, the optimizations that are introduced in this thesis are applicable to a large number of PS systems. To better understand the relation to the following chapters, a hint will be given that indicates at which place the later presented optimization can be plugged into the basic system.

As shown in Figure 2.1, the basic system consists of two parts: One part is running on the PS server, while the other part runs on the mobile device. The following details the execution steps of both parts separately.

3.1 Algorithm PS Server

Every time a sensing query is issued by a client to the query interface, the basic sensing system spawns a new instance of the query manager which is shown in Algorithm 1. Hence, many instances of this algorithm run on the PS server in parallel, if more than one query is processed at a time. One instance of the query manager is responsible for coordinating the processing of only one particular query. When the query result is received from a mobile device, the instance returns the sensing result back to the client and terminates.

Algorithm 1 Query manager for coordinating the processing of a query.

```
1:  procedure QueryManager(q)
2:      R ← M                                  ▷ Modified in optimized approach
3:      for all m ∈ R do
4:          sendQuery(m, q)
5:      end for
6:      result ← 0
7:      while (t_now < q.t_d) ∧ (result = 0) do
8:          result, m_r ← receiveResult()
9:      end while
10:     if result = 0 then
11:         return error
12:     end if
13:     for all m ∈ R\{m_r} do
14:         stopQueryMsg(m, q)
15:     end for
16:     return result
17: end procedure
```

After receiving query q, the query manager sends q to every mobile device $m \in M$, i.e., to every device that is located in the service area (Algorithm 1, lines 2–5). Next, the query manager waits until it receives the result of the query from at least one device, while constantly checking whether the sensing deadline has expired (lines 6–9). If the server does not receive any result before the sensing deadline t_d, the sensing query fails and the server returns an error

message to the query interface, which in turn informs the client (lines 10–12). As soon as the query manager receives sensor data from some mobile device, it sends a message to all devices that initially received q to stop the execution of the query. This avoids energy consuming redundant sensor readings in case other devices could afterwards also capture data for that query (lines 13–15). Subsequently, the received sensor data is returned back to the query interface, which forwards the query result to the client (line 16).

Note that the query manager distributes q to all $m \in M$. To optimize this simple distribution strategy, the efficient query distribution reduces R to only a subset of M that includes only those devices that promise high sensing success. The detailed description of the efficient query distribution is given in Chapter 5.

3.2 Algorithm Mobile Devices

For communicating with the PS server, each device runs the *query listener* that processes all messages that are received from the PS server. The processing steps of the query listener are shown in Algorithm 2.

Algorithm 2 Query listener for processing server messages.

```
1:  procedure QueryListener
2:      while TRUE do
3:          msg ← receive()                          ▷ Blocking call
4:          if msg instanceOf(queryMsg) then
5:              Q ← Q ∪ {msg.q}
6:          else if msg instanceOf(stopQueryMsg) then
7:              Q ← Q\{msg.q}
8:          end if
9:      end while
10: end procedure
```

To wait for an incoming message, the query listener executes a blocking call that returns a message once data is delivered to its process by the communication interface of the device (line 3). The query listener then identifies the type of the

message and adds the query that is contained in the message to the set of all known queries Q (lines 4–5) if the message is a query message. In contrast, if the message is a stop query message, it removes the query from Q (lines 6–7).

In parallel to the query listener, a device runs the *sensing engine* that is responsible for reading sensor data (see Algorithm 3). First, the device fixes its

Algorithm 3 Sensing engine for reading sensor data.

```
1:  procedure SensingEngine
2:      while TRUE do
3:          posInterval ← c_p                    ▷ Modified in optimized approach
4:          sleep(posInterval)
5:          p ← getPosition()
6:          for all q ∈ Q do
7:              if euclDist(p, q.p) ≤ q.range then
8:                  data ← readData()
9:                  sendToServer(data)
10:                 Q ← Q\{q}
11:             end if
12:         end for
13:     end while
14: end procedure
```

position by requesting data from the positioning sensor after having waited for time c_p (lines 3–5). Depending on whether the device moves outdoors or indoors this position is provided by the GPS sensor or an IPS, respectively. The value of c_p is technology dependent and is the minimum time frame for a sensor to provide a fresh position. For instance, GPS sensors can provide a new position every second [KLG+09], which is sufficiently accurate to not miss a sensing query when considering the typical movement speed of pedestrians. With this fresh position, the device checks if it is currently located in the sensing range of one of the queries in Q (lines 6–7). If that is true for a query q, the device reads data for q and uploads the data to the PS server (lines 8–9). Finally, it removes q from Q (line 10).

In this algorithm, a device periodically fixes its position (line 3). To reduce the

energy overhead for positioning, the efficient position sensing component changes this assignment to an individual time frame depending on the distance to the closest location where data is queried. A detailed description of the efficient positioning approach is given in Chapter 6 and Chapter 7.

4 Opportunistic Position Update Protocol

One major prerequisite to efficiently distribute sensing queries is that the PS server has knowledge about the geographical distribution of the mobile devices in its service area. Only with knowledge about this distribution, the server can estimate the sensing success of individual devices and decide which ones are promising candidates for receiving a given sensing query. To provide this information, this chapter introduces a position update protocol that runs on mobile devices and sends position information in an energy-efficient way to the PS server. In contrast to existing update protocols (see Leonhardi et al. [LR01] for an overview), this update protocol relies on so-called *opportunistic position updates*. To reduce energy overhead, these updates are sent together with other mobile device traffic back-to-back. The proposed approach significantly improves the energy efficiency compared to existing protocols and is therefore especially beneficial in PS systems.

In general, the application scope of the proposed position update protocol is not limited to a PS scenario. In fact, it can be applied to all scenarios in which a remote server keeps track of the positions of mobile objects. Typical examples are geo-social networks showing the current positions of friends or navigation services tracking smartphones to detect traffic conditions. To account for this fact, the proposed update protocol is described in this chapter in an application agnostic way. The connection between this update protocol and the other parts of the PS system is described in Chapter 5 in more detail.

The structure of this chapter is as follows: Section 4.1 contains a general motivation and overview, which points out the benefits and requirements for a position update protocol. Furthermore, it introduces the idea of using opportunistic updates. Section 4.2 introduces a detailed energy model for mobile

communication that will be used as reference model for measuring communication energy throughout this thesis. Next, the problem statement in Section 4.3 formalizes the goal of the update protocol, before the results of a communication study are discussed in Section 4.4. The goal of this study is to show the feasibility of the opportunistic approach by analyzing communication events in everyday smartphone usage. Section 4.5 presents the design and implementation of the opportunistic position update protocol that is based on a Markov decision process. To show the efficiency of this protocol, the evaluation results are presented in Section 4.6. Finally, we have a look on related work in position update protocols in Section 4.7 before this chapter is concluded in Section 4.8.

4.1 Motivation and Overview

Tracking the positions of mobile objects by a remote server is a problem that is relevant in many applications. For instance, consider a friend finder that lists all friends of a user who are within a certain distance. Apparently, the positions of all friends must be known to the system in order to determine which friends are close to the user. One other example is a fleet management system in which a control system tracks and visualizes the movement of vehicles. In order to implement such location-based applications (LBAs) efficiently, location servers (LS) are used that manage mobile object positions and are deployed in the infrastructure (see Figure 4.1). Typical examples for such systems are Google Latitude [Inc] or Trace4You [Gmb]. The basic principle is straightforward: Instead of sending its position to each LBA individually, the mobile object updates its position on the remote LS. This server provides LBAs with mobile object positions by implementing spatial query and eventing functionalities. For instance, the friend finder only needs to query the LS for the positions of the friends instead of querying the device of each friend individually for its position. Hence, a LS relieves a mobile object from sending the same position to different LBAs multiple times and is therefore also a prerequisite for advanced LBAs that need to access positions from many mobile objects.

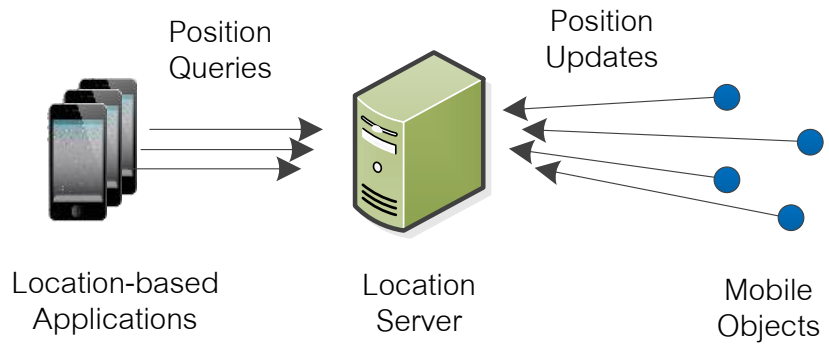


Figure 4.1: Location-based applications querying position data that is provided by a location server.

One big concern in tracking mobile objects is the energy that is required by a device for sending its positions to the LS, since mobile data communication typically induces a substantial energy overhead [SSM09; BDR12b]. Obviously, this severely impacts the user's acceptance to use such LBAs. Therefore, several *position update protocols* have been investigated in the literature [LR01] that try to minimize the communication overhead while guaranteeing a certain quality of position information (e.g., accuracy) managed by the LS. Typical strategies include *time-based* update protocols that restrict the update rate, *distance-based* protocols sending an update whenever the mobile object has moved a certain distance, or *dead reckoning* using movement prediction functions at the mobile object and LS to only send updates whenever the predictions deviate by more than a given accuracy threshold.

Although these are effective strategies in general, these protocols have one major shortcoming: They abstract away from the properties of the network interface by assuming that every position update message consumes the same amount of energy. However, as recent studies have shown, this assumption does not hold true for the cellular network interfaces of current mobile devices [BBV09; LZZ11; DB12]. Typically, these interfaces implement their own power management which interferes with the update protocol. In more detail, after sending a message, the network interface is not immediately going into power-save mode again but rather stays in high-power mode for a certain time span

called *tail time*, waiting for further messages to be sent. Therefore, sending messages in batches is more energy efficient since the energy that is spent in the tail time is minimized.

Obviously, these specific characteristics of cellular network interfaces have to be taken into account in the design of position update protocols. To this end, the proposed update protocol introduces an extension to existing position update protocols—namely time-based, distance-based, and dead reckoning—that schedule the transmission of position updates to minimize the energy of communication. The basic idea is to send position updates in an opportunistic manner, i.e., position updates are sent "back-to-back" after messages from other applications to minimize the overall tail time. However, it would be inefficient to send a position update with *every* outgoing message since this might increase the number of update messages beyond the necessary limit required to guarantee a certain position quality at the LS. Therefore, this chapter introduces an algorithm that tries to find both, the optimal time to send an update as well as the minimal number of update messages under the constraint that a certain quality of position information has to be achieved. Obviously, this requires an online optimization strategy since the points in time when messages from other applications will be sent are unknown a priori. To this end, a Markov decision process is used to estimate the amount of energy that can be saved by sending an update together with another message under the current communication pattern.

4.2 Energy Model for Mobile Communication

In general, the LS and mobile devices communicate via a cellular network. To evaluate the energy consumption of an update protocol on the mobile device, the energy profile of the device's cellular communication interface is characterized by following the empirical results given in [BBV09; LZZ11; DB12]. These studies agree that the consumed energy for sending messages does not only depend on the size of the messages but also on the time that passes between the

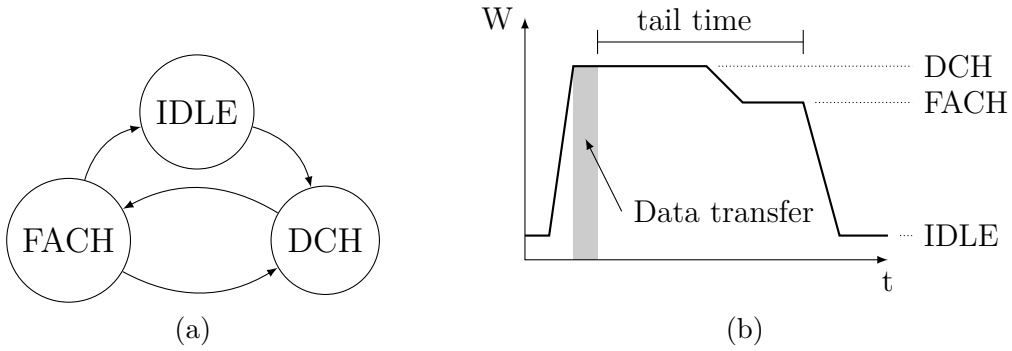


Figure 4.2: Energy characteristics of the cellular communication: (a) The power states of the interface. (b) Power consumption for a data transfer over time.

transmissions of messages. The reason for this non-linear energy characteristic is that a cellular network interface has different power states with different state transition times. As a result, the time the interface stays in a certain state depends on the temporal pattern of the send and receive events of the device. In the following, these states and their transitions are briefly introduced.

A cellular network interface has an idle state (*IDLE*) and two operating states (*DCH* and *FACH*) (see Figure 4.2a [QWG+10a; LZZ11]). The interface switches from *IDLE* to the high power state *DCH* once data is transmitted. When the transmission is finished, the interface does not immediately switch back to *IDLE* (see Figure 4.2b). Instead, it switches after some seconds to the slightly less power consuming *FACH* state and stays there for some additional seconds, before it switches to the power saving *IDLE* state again. As a result, after a transmission is finished the interface stays in high-power mode (*DCH* and *FACH*) for some additional time (around 10 s). This time is referred to as *tail time* and the energy that is consumed in this time frame is referred to as *tail energy*. The purpose of the tail time is to keep the interface powered up for some time to avoid that it has to be powered up again for each consecutive data transfer, which would cause an energy overhead and additional delay. However, this behavior implies that even small data transfers consume a significant amount

of energy, since the interface stays powered up for the tail time.

For the rest of this chapter, we denote the sending of a position update message at time t as u_t. The energy consumed on the mobile device for sending a message u_t is denoted as $E(u_t)$. The energy measurement shown in Figure 4.2b indicates that sending a position update right after another transmission can be beneficial in terms of energy. In contrast, if we look at position updates in an isolated fashion, the given energy characteristic is disadvantageous. Since the size of a position update message is typically only a few kilobytes including the protocol overhead, most of the energy for sending such a message is consumed as tail energy. Therefore, substantial energy savings can be achieved if a position update is sent "back-to-back" right after the start of the tail time of another transmission on the device.

4.3 Problem Statement

The problem that needs to be tackled for implementing an opportunistic update protocol can be defined as a constrained optimization problem: The energy spent for position updates should be minimized while guaranteeing a certain quality of the mobile object position stored on the remote LS. Here, the quality is defined by the *quality constraint* of the position update protocol. For instance, a time-based update protocol requires that the position stored on the LS must not be older than a certain maximum age. More formally, for two consecutive position updates at times t_i and t_{i+1}, respectively, $t_{i+1} - t_i$ must be less than or at most equal to the threshold δ_T. For a distance-based update protocol, the distance between the position stored on the LS and the current mobile object position must be at most a given maximum distance δ_D.

As apparent from the description of the communication energy model, the energy overhead for sending update messages depends on the time when the updates are sent in relation to the time when other messages are sent. Let $T = (t_1, \ldots, t_n)$ be the sequence of timestamps when position updates are sent.

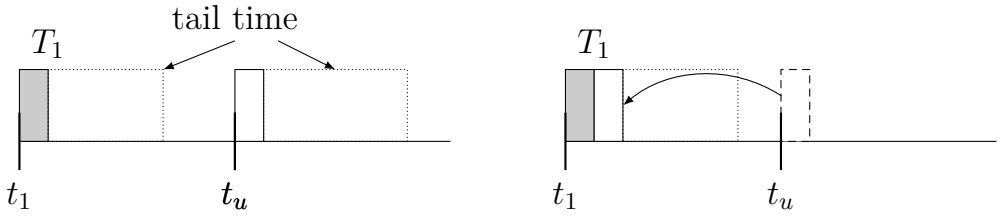
Then the constrained optimization problem can be formulated as follows:

minimize	$\sum_{t_i \in T} E(u_{t_i})$
subject to:	Quality constraint fulfilled
variables:	$T = (t_1, \ldots, t_n)$

That is, the goal is to find an optimal sequence of update times that leads to minimal energy consumption under the constraint that the quality constraint of the update protocol must not be violated.

Note that in the sequence $(t_1, \ldots, t_n)$ the point in time t_i of each update message u_{t_i}, as well as the number n of update messages, are variables. The number of update messages increases by sending updates earlier than actually required by the quality constraint to utilize the uptime of the network interface caused by other messages. Such an early update is called *opportunistic update*, since the opportunity is taken to send a position update right after another message. In contrast, *forced updates* are updates that are required to fulfill the quality constraint, for instance, in periods when no other messages are sent by other applications. Sending opportunistic updates with every other message and sending only forced updates are two extremes in a spectrum of possibly update sequences. Also note that this optimization problem has to be solved online without knowing the transmission times of future messages.

To illustrate this problem, consider the following scenario: The communication profile of a device is given in Figure 4.3a. This profile shows a filled rectangle that indicates a transmission and a dotted box that represents the tail time until the interface powers down. In this scenario, a device executes a time-based position update protocol which requires that the next position update is sent not later than time t_u. At time t_1, some application on the mobile device transmits data over the network interface (transmission T_1). Sending the position update opportunistically right after T_1 is more efficient than sending the update at time t_u, since the tail energy can be reduced by almost 50% (see Figure 4.3b).



(a) Forced update (b) Opportunistic update

Figure 4.3: Sending a position update: Scenario 1.

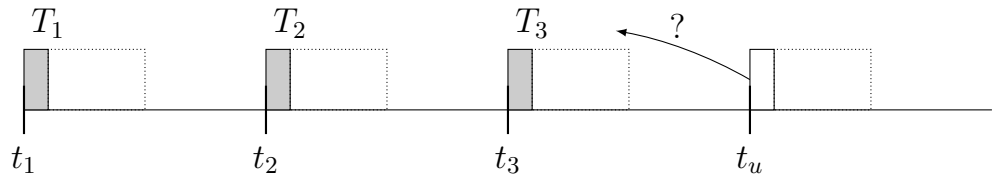


Figure 4.4: Sending a position update: Scenario 2.

However, the scenario depicted in Figure 4.4 shows that sending opportunistic updates with every outgoing message might be less energy efficient than sending forced updates. In this scenario, three transmissions T_1, T_2 and T_3 take place before the position update is due. In this case, sending the position update after T_3 would be the best solution, since T_3 is closest to t_u. However, at time t_1 it is not clear whether another transmission will take place before the quality constraint is violated at time t_u.

These examples illustrate the challenge to find the optimal update sequence, which in general corresponds to a mixture of opportunistic and forced updates, without knowing transmission times a priori.

Before we look at the actual approach to tackle this online optimization problem, the next section shows that realistic communication patterns are suitable for opportunistic updates. This is done by presenting the results of a communication study that was conducted in the context of this thesis.

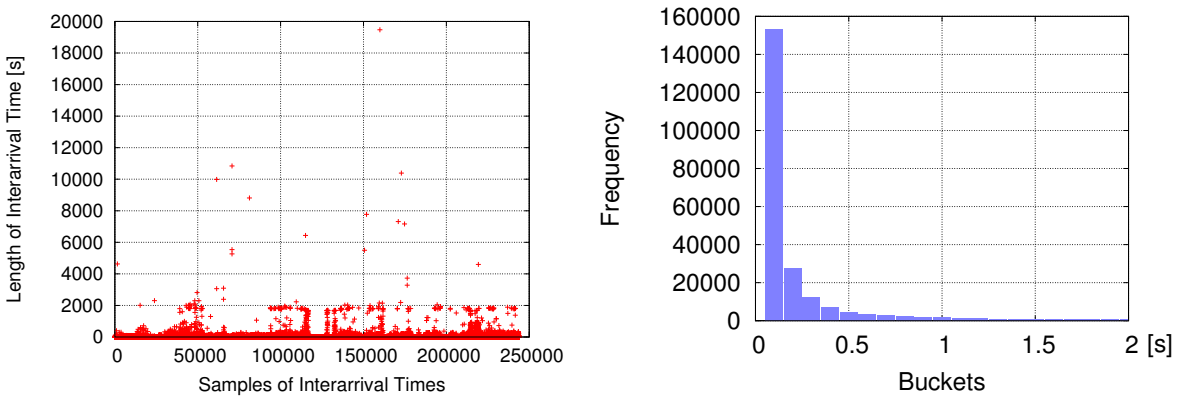
4.4 Communication Study

A necessary prerequisite for the opportunistic update approach is that a device communicates frequently in order to provide opportunities to send opportunistic position updates. For instance, if a device communicates only once every hour, it would have to send mostly forced updates when using a time-based protocol with $\delta_T < 1\,\text{h}$. Hence, to show that this approach is feasible for the typical communication pattern of a mobile device, we have a look on the results of an empirical study that was conducted in the context of this work.[1]

For this study, an Android application was developed that tracks the time between two consecutive transmissions on a mobile device. To this end, the application monitors the cellular network interface with a sampling interval of 1 ms and checks whether data has been sent or received (using the *TrafficStats* class of the Android API). The time intervals between two consecutive transmission events (denoted as traffic interarrival times) are logged to the SD-card to be analyzed offline. For this study, traffic traces from 12 different users were collected that executed this application in the background while using their Android phone in their everyday lives. Most of these smartphones run typical applications that are installed on the majority of modern smartphones, such as a mail client, a web browser, or an instant messenger (but no position update protocol). In total, the traces included over 500 hours of smartphone traffic.

To average these results over all users, two days of the communication trace from each user were extracted, which is the length of the shortest trace that was collected. Figure 4.5a shows the scatter plot of all traffic interarrival times that were obtained by this study with their respective length on the y-axis. It shows that the vast majority of the interarrival times are not longer than a few seconds. Except for some outliers, all interarrival times are smaller than 2000 s. To visualize the frequency of different interarrival times, they were classified with a histogram of bucket size of 0.1 s (see Figure 4.5b). Obviously, the distribution of

[1] The communication traces obtained by this study are available on the website: `http://www.comnsense.de/downloads/`



(a) Scatter plot (b) Histogram

Figure 4.5: Result of the study to measure traffic interarrival times.

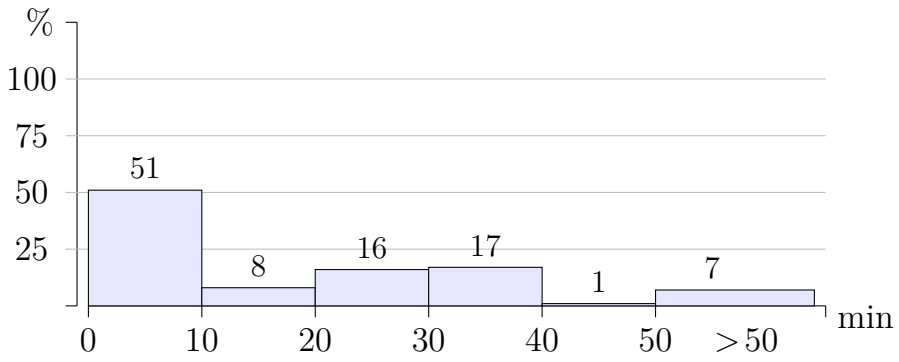


Figure 4.6: Percentage of interarrival times with respect to the total time.

the interarrival times follows an exponential distribution, i.e., short interarrival times occur very frequently. For instance, the interarrival times that are smaller than one second account for more than 89% of all interarrival times. This can be attributed to the fact that an interaction between a mobile device and a remote server is in most cases characterized by many interactions which are temporally close, like downloading a web page.

Although larger interarrival times are unlikely, their occurrence would mean that there is no possibility to send opportunistic updates. Therefore, it would be interesting to check what fraction of the time these devices spend in large interarrival times. To this end, the interarrival times were categorized according

to their length in bins of size 10 minutes. For every interval $[10i; 10(i+1)[$ with $i \geq 0$ the sum of all interarrival times that lie in this interval was calculated. Figure 4.6 shows a plot of the fraction of this sum with respect to the total sum of all interarrival times in a histogram. The results show that in 51 % of the time the interarrival time was smaller than 10 minutes. Moreover, in 92 % of the time the interarrival time was smaller than 40 minutes.

This study concludes that it is very likely that a device communicates within a time frame of 40 minutes at least once. Moreover, in more than half of the cases, the time gap between two consecutive transmissions is smaller than 10 minutes. Hence, it is very likely that an opportunistic update can be sent at least every 10 minutes.

4.5 Opportunistic Update Protocols

This section presents the opportunistic extensions for position update protocols. It starts by providing an overview that shows how the opportunistic update strategy can be integrated with existing position update protocols. Subsequently, a Markov decision process is introduced that decides online for a given transmission whether an opportunistic position update should be sent by the mobile device. Finally, it is shown how to integrate three existing update protocols with the opportunistic approach, namely, time-based, distance-based, and dead reckoning protocols.

4.5.1 Overview

The basic idea of integrating opportunistic updates into existing position update protocols is as follows: Two algorithms run in parallel on the mobile device. On the one hand, the device executes the *basic (original) position update protocol* (see Algorithm 4) that ensures that the quality constraint of the update protocol is not violated. More precisely, whenever the protocol specific update condition is fulfilled, the basic update protocol sends a forced update to guarantee the quality of position information managed by the remote LS. The update condition

is derived by the given quality constraint and will be explicitly defined later on.

Algorithm 4 Basic version of the position update protocol.

```
1: procedure basicUpdateProtocol
2:     while true do
3:         p ← getPosition()
4:         if checkUpdateCondition() then
5:             sendPositionUpdate(p)
6:         end if
7:     end while
8: end procedure
```

On the other hand, the device executes the opportunistic update protocol which sends opportunistic updates together with other messages (see Algorithm 5). In order to minimize the energy consumption, this protocol decides if it is beneficial to send an opportunistic update whenever another message is transmitted by the device (line 5). The basic difficulty is to estimate online whether another transmission will take place before the time when an position update has to be sent according to the quality constraint of the protocol. In this case, the position update should be deferred at least until the next transmission, otherwise, the update should be sent with the current transmission. In the next section, a Markov decision process is described that decides whether it is beneficial to send an opportunistic update based on the prediction of message transmissions.

Algorithm 5 Opportunistic extension of the position update protocol.

```
1: procedure updateProtocolExtension
2:     while true do
3:         waitForNextTransmission()
4:         p ← getPosition()
5:         if checkOpportunisticUpdate() then
6:             sendPositionUpdate(p)
7:         end if
8:     end while
9: end procedure
```

4.5.2 Markov Decision Process

To decide whether an opportunistic position update should be sent with the current transmission or not, a Markov decision process (MDP) [Bel57] is used. An MDP is an extension of a Markov chain that additionally incorporates actions that can be executed in states and costs (or rewards) that are associated with state transitions. Given a concrete implementation of an MDP, one action can be determined for each state that minimizes the sum of the expected costs. Including transition costs and uncertainty about future states, an MDP incorporates all aspects of our decision problem.

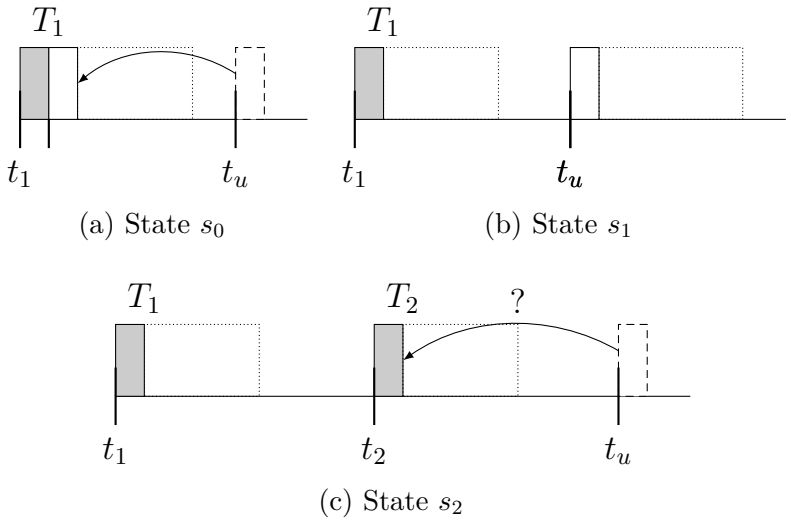
In the following, the basic states of the MDP are introduced, before the transition probabilities between these states are defined. To complete the definition of the model, transition costs are assigned that reflect the energy consumption of a mobile device. Finally, it will be shown how the MDP can be used to decide whether an opportunistic update should be send.

4.5.2.1 Structure of the MDP

To map the aforementioned problem to an MDP, a generic state s_d is constructed that represents the starting point of the decision. From this state two possible actions are available, which are either to send an opportunistic position update with a given transmission or not. Hence, the possible actions that can be taken in state s_d are defined as $A = \{send, \overline{send}\}$. The relation to Algorithm 5 is as follows: Every time Algorithm 5 reaches line 5, the MDP is in state s_d and decides whether to take action *send* or $\overline{send}$. The result of this decision is then returned back to Algorithm 5.

Starting from state s_d and choosing one of the two available actions in A, the following events can occur:

1. Action *send* is taken and an opportunistic position update is sent right after the current transmission (see Figure 4.7a). To describe this event, state s_0 is introduced.



(a) State s_0 (b) State s_1

(c) State s_2

Figure 4.7: States of the Markov decision process.

2. Action $\overline{send}$ is taken and no other transmission will occur before the quality constraint is violated (see Figure 4.7b). In this case, the position update cannot be sent opportunistically, since there is no further transmission available before t_u. As a result, the position update will be forced at time t_u. To describe this event, state s_1 is introduced.

3. Action $\overline{send}$ is taken and at least one other transmission will occur on the device before t_u (see Figure 4.7c). In this case, the sending of the opportunistic update can be postponed until the next transmission. To describe this event, state s_2 is introduced.

Figure 4.8 gives an overview of the states and transitions of the MDP. Note that for defining these states it is assumed that the update due time t_u is known, i.e., the time when the quality constraint will be violated. It will be shown later on, how t_u can be predicted for the different update protocols. Next, the transition probabilities for the state transitions are defined.

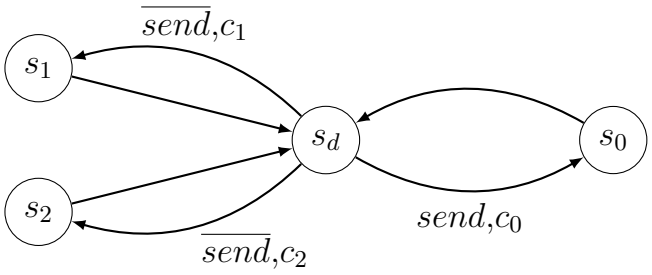


Figure 4.8: States and transitions of the Markov decision process.

4.5.2.2 Transition Probabilities

To define the transition probability from state s_d to state s_2, we need to quantify the probability that another transmission occurs on the device before t_u. For this purpose, the random variable X_t is introduced that describes the arrival time of the next transmission. The random variable X_{t-1} is referred to as the last traffic interarrival time which follows the same distribution as X_t. Since the last interarrival time τ_{t-1} can be determined by the device, this knowledge is utilized to define $p_{\overline{send}}(s_d, s_2)$ as the following conditional probability:

$$p_{\overline{send}}(s_d, s_2) = P(X_t \leq t_u | X_{t-1} = \tau_{t-1})$$

We can then define the probability that no other transmission arrives before t_u as:

$$p_{\overline{send}}(s_d, s_1) = P(X_t > t_u | X_{t-1} = \tau_{t-1})$$

It is easy to see that these probabilities sum up to 1. For all other transitions (e.g., $p_{send}(s_d, s_0)$), the transition probability is 1. As a result, after deciding on an action, the MDP automatically returns to state s_d.

Since the MDP is executed on the mobile device, the device has to know the distribution of its traffic interarrival times. For this purpose, the device can either be provided with a typical distribution of X_t or it can learn this distribution over time by logging its traffic interarrival times as demonstrated by the communication study in Section 4.4.

4.5.2.3 Cost Model

To finish the definition of the MDP, transition costs are assigned in order to be able to measure the effects when deciding on an action. Since the overall goal is to minimize the energy consumption of the mobile device, the transition costs of the MDP are related to the energy costs that result from moving to the different states. Cost c_1 denotes the energy cost to move from state s_d to state s_1 and is defined as the energy that is consumed for sending a forced update including its full tail energy. This energy is denoted as E_u. Moving to state s_2 does not induce any costs since no message is sent, instead it will be waited for the next transmission that occurs before t_u. Hence, c_2 is set to be zero. To define the energy costs c_0 when moving to state s_0, we need to quantify the energy costs that result from sending an opportunistic position update. In general, sending a position update earlier than required by the quality constraint can increase the number of position updates that need to be sent in total. However, the closer an opportunistic update is sent to the original update due time t_u, the smaller are the chances that additional updates have to be sent. To map this risk to a cost measure, an early sending is associated with the proportional costs for sending a forced position update. More precisely, assuming that the last position update was sent at time t_{u-1} and the next will be due at time t_u, the cost for sending a position update already at time t is given as $E_u \cdot (t_u - t)/(t_u - t_{u-1})$.

With these definitions, the cost model is defined as follows:

$$
\begin{aligned}
c_0 &= E_u \cdot (t_u - t)/(t_u - t_{u-1}) \\
c_1 &= E_u \\
c_2 &= 0
\end{aligned}
$$

With this cost model, the definition of the MDP is now complete. Next, we look on how it can be used for deciding which action should be taken in state s_d.

4.5.2.4 Deciding on an Action

Given the definition of the MDP, we can calculate which action in state s_d results in the lowest expected costs (referred to as the *optimal* action). In general, the computational effort of finding the optimal action depends on the value of the discount factor $\gamma \in [0; 1)$, which is given as a parameter of the MDP. This discount factor determines how many future states are taken into consideration when evaluating the costs of a given action. Consequently, the smaller γ is chosen, the more efficient can the optimal action be computed. For instance, $\gamma = 0$ takes only the cost for the next set of actions into account, while the extreme case of $\gamma = 1$ would take the infinite number of all future states into account. Since the overall goal is to reduce the energy consumption of a mobile device, this computational effort has to be kept as low as possible. Hence, γ is set to zero. This means that no expected costs of future states are considered. This may on the one hand lead to less accurate predictions, however, on the other hand the optimal action a^* for a state s can be found very efficiently by solving the following optimization function:

$$a^* = \arg\min_{a \in A} \Big(\sum_{s'} p_a(s, s') \cdot c_a(s, s') \Big)$$

Here, p_a denotes the state transition probability when choosing action a and c_a the costs that are assigned to this transition. Since only two actions need to be taken into account, this equation can be formulated by the following closed formula:

$$\begin{aligned} & \overbrace{E_u \cdot (t_u - t)/(t_u - t_{u-1})}^{\text{Action } send} < \overbrace{p(s_d, s_1) \cdot E_u}^{\text{Action } \overline{send}} \\ \Leftrightarrow\ & P(X_t \leq t_u | X_{t-1} = \tau_{t-1}) < \frac{t - t_{u-1}}{t_u - t_{u-1}} \end{aligned} \tag{4.1}$$

If this inequality is fulfilled, the expected energy costs for sending an opportunistic position update are less than the cost of not sending an opportunistic update. In this case an opportunistic position update is sent.

However, to solve Equation 4.1 we need knowledge about the due time t_u of the next position update. Next, we will see how to calculate or estimate this value for the time-, distance-based and dead reckoning position update protocol.

4.5.3 Time-based Update Protocol

As mentioned earlier, the time-based update protocol forces a position update to the LS in a predefined time interval δ_T. More precisely, the update condition is fulfilled at time t if the following inequality holds, where t_{u-1} denotes the time when the last position update was sent:

$$t - t_{u-1} \geq \delta_T$$

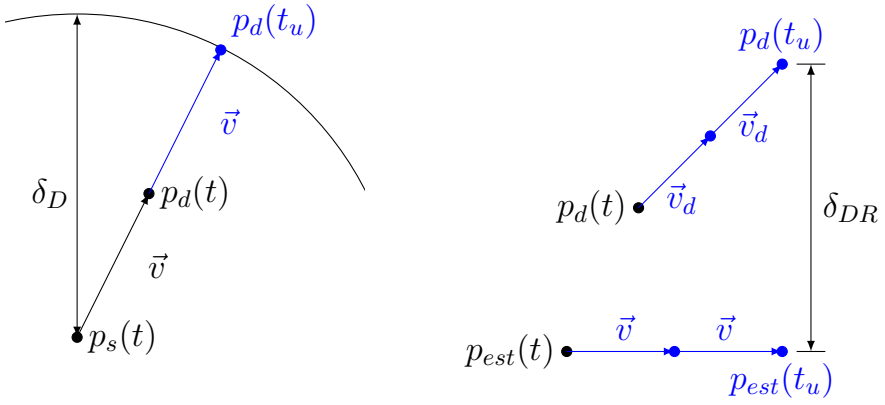
For this protocol, all parameters to evaluate Equation 4.1 are known, since time t_u can be easily calculated based on the time of the last update.

4.5.4 Distance-based Update Protocol

The update condition of the distance-based update protocol is defined by the Euclidean distance between the current position of the mobile device $p_d(t)$ and the last updated position $p_s(t)$ on the LS. When this distance at time t is bigger than accuracy threshold δ_D, a position update is sent:

$$d_{Eucl}(p_d(t), p_s(t)) \geq \delta_D$$

Applying the MDP to this protocol brings up the difficulty that time t_u, at which the next position update is due, is not known at decision time. Since this value is needed to evaluate Equation 4.1, it is predicted by the time at which the quality constraint is likely to be violated if the device keeps its current movement direction and speed. To this end, the last movement trajectory $\vec{v}$ of the device is used and it is assumed that the device keeps on moving according this vector (see Figure 4.9a). The time until the next update can then be predicted by value Δt for which the distance between $p_d(t) + \Delta t \cdot \vec{v}$ and $p_s(t)$ is equal to δ_D.



(a) Distance-based (b) Dead reckoning

Figure 4.9: Prediction for the time of the next position update t_u.

4.5.5 Dead Reckoning Update Protocol

Similar to the distance-based protocol, the dead reckoning update protocol defines a threshold on the deviation between the position on the device and the LS. However, the position on the LS is not statically set to the position that was contained in the last position update, but evolves over time. For this purpose, the device includes a movement estimation trajectory $\vec{v}$ with each position update. The LS can now estimate the device position $p_{est}(t)$ at time t as follows:

$$p_{est}(t) = p_{u-1} + \vec{v}(t - t_{u-1})$$

Here, p_{u-1} and t_{u-1} refer to the position and time of the last received position update. The protocol triggers a position update at time t if the device position deviates more than threshold δ_{DR} from the estimation of the LS:

$$d_{Eucl}(p_d(t), p_{est}(t)) \geq \delta_{DR}$$

As for the distance-based protocol, update time t_u needs to be predicted. However, not only the device position but also the position stored on the LS is changing over time. Therefore, two vectors need to be considered for the pre-

diction: The actual movement vector of the device, denoted as $\vec{v}_d$, and the movement vector $\vec{v}$ stored on the LS which was updated with the last position update, respectively. To predict time t_u, an extrapolation is needed for the current device position $p_d(t)$ and the estimated position $p_{est}(t)$ on the LS with the respective vectors (see Figure 4.9b). The time until the next update can then be predicted by the value of Δt for which the distance between $p_d(t) + \Delta t \cdot \vec{v}_d$ and $p_{est}(t) + \Delta t \cdot \vec{v}$ is equal to the accuracy threshold δ_{DR}.

4.6 Evaluation

In this section we look at the evaluation results for the proposed opportunistic update protocols. First, the experimental setup of the evaluation is introduced, before the evaluation results are presented.

4.6.1 Experimental Setup

To evaluate the efficiency of an update protocol, the communication energy for sending position updates needs to be measured. However, accurate energy measurements on mobile devices require the deployment of additional hardware [RH10] and make long term measurements for scenarios that include device mobility hardly feasible. To be able to inspect a large variety of scenarios with different parameter settings, a software tool was implemented that simulates the behavior and characteristics of a mobile device. The simulated device runs an update protocol and sends position updates via a cellular network interface. To model this interface, an uplink speed of 384 kpbs [RHS+06] is assumed and the energy model of Liu et al. [LZZ11] is used to model the energy for sending messages via the 3G network. This energy model was empirically derived by measurements on a HTC G1 smartphone. The power consumptions for the different states of the interface are shown in Table 4.1. Note that these energy characteristics were verified by several other measurements that obtained similar results [BBV09; DB12].

To consider the communication of other applications on the device, the communication traces that were collected for the communication study (see Section 4.4) were used to simulate traffic on the device. To model the movement trajectory of the mobile device, real-world GPS traces of persons either walking or running were used that were extracted from OpenStreetMap [OSM]. The typical GPS sampling interval of these traces lies between one and three seconds.

State	Power	State Transition	Delay
DCH	570 mW	IDLE → DCH	2 s
FACH	401 mW	DCH → FACH	4 s
IDLE	10 mW	FACH → IDLE	6 s

Table 4.1: Energy model for 3G communication.

The evaluation compares the following three approaches:

basic The basic (non-opportunistic) versions of time-based, distance-based, and dead reckoning update protocols.

extend The update protocol that uses the opportunistic extension that is presented in this chapter.

nopred In addition to these two approaches, the evaluation shows the results of one additional approach that implements a generic (non-predictive) opportunistic update protocol. In contrast to the opportunistic position update protocols, this non-predictive protocol sends a position update with *every* message from other applications. That is, it does not include the MDP to predict future traffic and estimate the benefit of sending or deferring position updates (line 5 in Algorithm 5 evaluates always to true). On the one hand, the evaluation of this approach shows the performance of a simple protocol that uses opportunistic updates in comparison to the basic approach. On the other hand, the results show the additional benefits of the

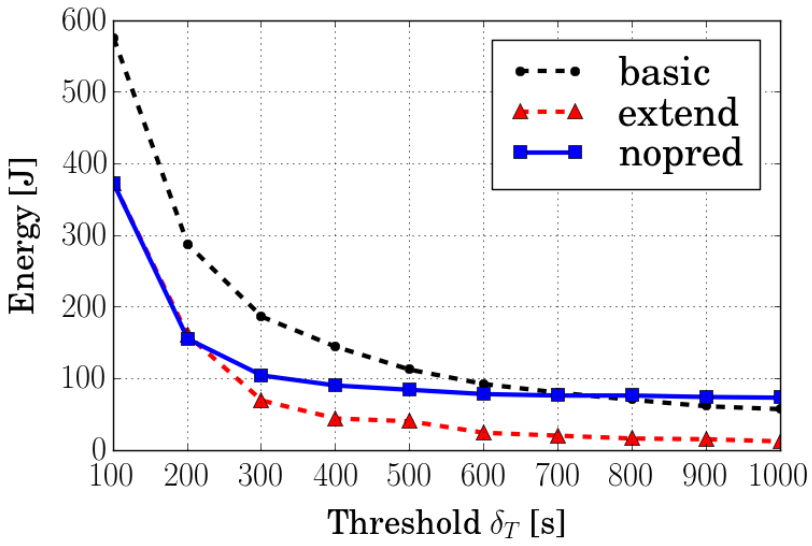


Figure 4.10: Energy consumption for the time-based update protocol.

extend-approach that uses an MDP for deciding when to send opportunistic updates.

To obtain the following results, each protocol was simulated for a time period of three hours.

4.6.2 Energy Consumption

To evaluate the energy efficiency of an update protocol, we only consider the energy that is consumed for sending position updates to the LS. However, since other applications also power up the communication interface, this value cannot be determined from a single simulation run. Hence, in a first step, the energy consumption of a device was measured when not using any update protocol. This value depicts the amount of energy that is consumed by other applications on the device and is denoted as E_{app}. Subsequently, the energy consumption of the device when using one of the update protocols is evaluated and denoted as E_{total}. The energy that is consumed solely by the update protocol is then given as $E_u = E_{total} - E_{app}$. Figures 4.10, 4.11 and 4.12 show E_u for the three protocols using different update thresholds. For the time-based protocol (Fig-

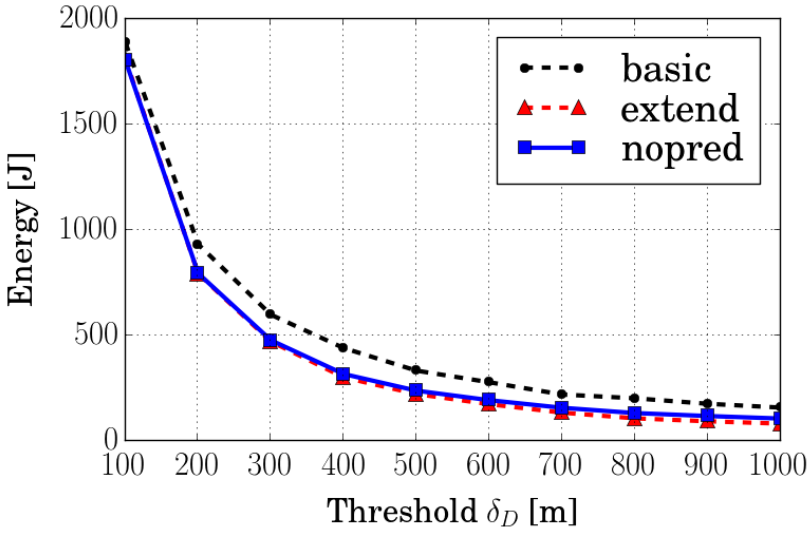


Figure 4.11: Energy consumption for the distance-based update protocol.

ure 4.10), the opportunistic approach consumes on average 70% less energy than the basic approach, while the non-predictive approach consumes in some cases even more energy than the basic protocol. The gap to the basic approach is getting larger with smaller values of update interval δ_T. Obviously, the more updates the protocol requires, the more energy can be saved by the opportunistic approach. Since the non-predictive approach triggers a position update with each transmission on the device, this approach is efficient if the number of required updates is high, i.e., for small values of δ_T. However, with larger values of δ_T, this approach gets inefficient since too much energy is consumed for sending unnecessary position updates. In contrast, the opportunistic approach is also efficient for large values of δ_T. This shows that the MDP chooses the right point in time to send an opportunistic position update. Hence, opportunistic position updates are sent with only a small subset of transmissions.

For the distance-based (Figure 4.11) and the dead reckoning (Figure 4.12) update protocols, we can observe similar results. Using the distance-based protocol, the opportunistic approach performs on average 35% better than the basic approach and 17% better than the non-predictive approach. For the dead reckoning protocol, the opportunistic improves energy efficiency on average by 17%

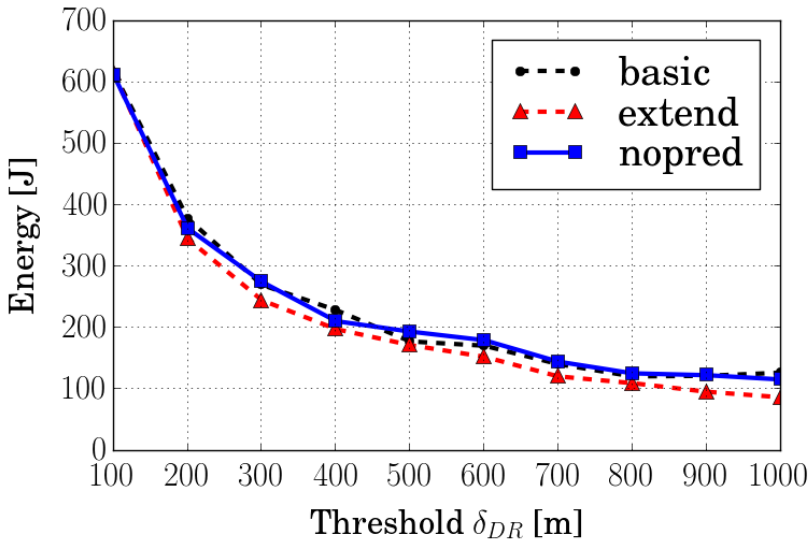


Figure 4.12: Energy consumption for the DR-based update protocol.

compared to the basic approach, while the non-predictive approach consumes almost the same amount of energy as the basic protocol. For the same reasons as explained before, the difference between the opportunistic approach and the basic approach decreases with a larger update threshold, while the gap between the opportunistic approach and the non-predictive approach increases.

Compared to the time- and distance-based protocol, the relative energy savings for the dead reckoning update protocol are the least. To better understand this behavior, Figure 4.13 directly compares the distance-based and the dead reckoning protocol to each other. We can see that the dead reckoning update protocol outperforms the distance-based protocol even in the basic version. Hence, the use of movement predictions obviously reduces the number of required position updates significantly. As a result, the benefit that is gained from the opportunistic approach is smaller because there is not much room for improvement. For a larger update threshold, the advantage of the dead reckoning protocol slowly decreases. However, we can see that the opportunistic extensions reduce the energy consumption for all parameter settings.

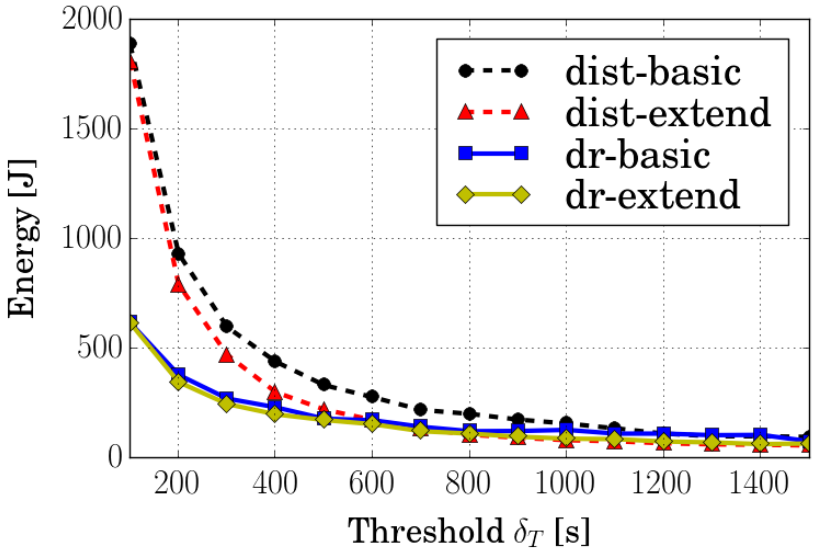


Figure 4.13: Comparison of energy consumption for different protocols.

4.6.3 Update Messages

To understand the results from the energy evaluation in more detail, we have a look at the number of position updates that are sent. Figure 4.14 shows this number for each approach with a fixed update threshold of 500 s (time-based protocol), respectively 500 m (distance-based and dead reckoning protocols). Moreover, the figure also includes the fraction of position updates that were sent opportunistically, i.e., during the tail time of other transmissions. Note that the results of the time- and distance-based protocols are not comparable with each other, even if they use a similar update threshold. The number of position updates in the distance-based protocol are also dependent on the movement pattern of the mobile device, which is not true for the time-based protocol. Hence, in the following we are only interested in the relative performance of one protocol regarding the different approaches to implement it.

The basic approach sends for each update protocol the fewest number of position updates, since an update is only sent when the quality constraint is going to be violated. Therefore, this approach can be considered as a lower bound on the number of position updates. We see that even in the basic approach a

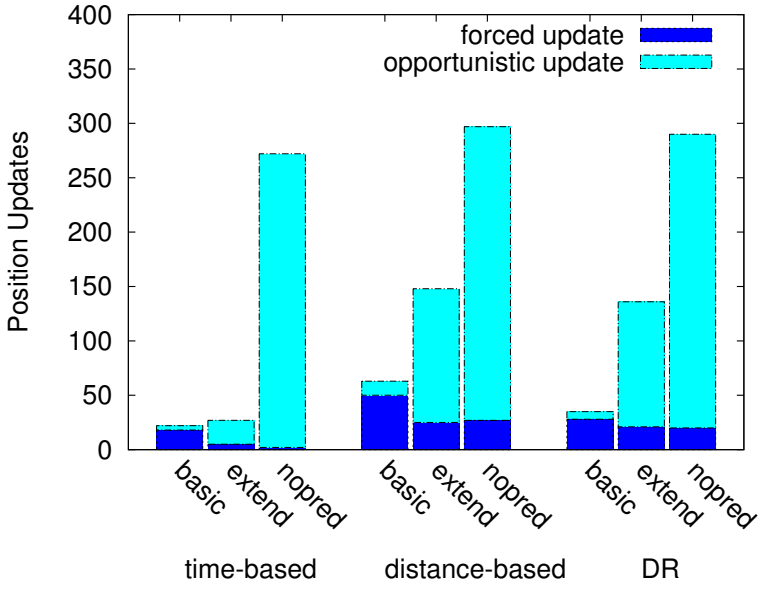


Figure 4.14: Number of position updates that were sent.

small fraction of position updates are coincidentally sent in the tail time of other transmissions.

The non-predictive approach sends the most position updates. The opportunistic approach can be considered as tradeoff between these two extremes. While the number of position updates increases compared to the basic approach, a large fraction of messages are sent opportunistically and, thus, sent in an energy efficient manner. For the time-based protocol, for which the next update time is known, the opportunistic approach manages to send almost all update messages opportunistically, which results in large energy savings as we have seen before.

4.6.4 Accuracy

Finally, we take a closer look at the position accuracy that the distance-based and dead reckoning protocol provide. Although the target accuracy of these protocols is defined by their respective quality constraint, it can be of interest how big the actual deviation between the device position and the position stored

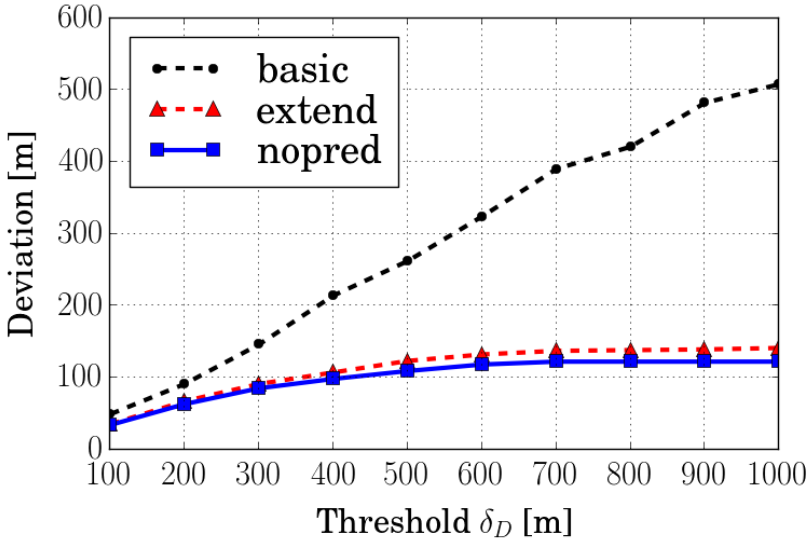


Figure 4.15: Position accuracy of the distance-based protocol.

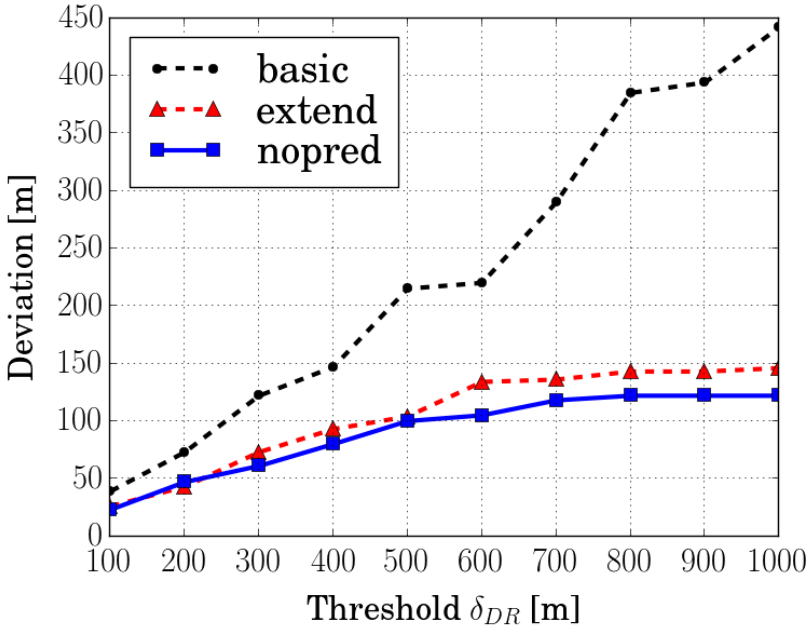


Figure 4.16: Position accuracy of the dead reckoning protocol.

on the LS is on average. For this purpose, for each time $t \in T$, where T is the total simulation time, the Euclidean distance between the current device position and the position stored on the LS is determined. Figures 4.15 and 4.16 shows the average of this deviation for different accuracy thresholds.

As we have seen before, the extended protocol and the non-predictive protocol trigger in total more position updates than the basic update protocol. As a result, the position of the mobile device is updated more often on the LS and the deviation between device position and LS position decreases on average. Hence, the opportunistic extensions not only decrease the energy consumption of the basic update protocols but also increase the position accuracy on the LS.

4.7 Related Work

Before this chapter is concluded, we have a look on related work considering position update protocols and energy-efficient communication. As already mentioned, existing work on update protocols [LR01; WXX+11; FHR12; Kjr12] aims to minimize the number of position updates without considering the specific energy characteristics of the communication interface. The opportunistic update protocol that was presented in this chapter extends the three prevailing update protocols proposed in [LR01].

Within the last few years, the awareness to also consider the power characteristics of mobile devices in algorithm design increased [PJH+12; PHZ12; PMR14]. Considering communication energy, different empirical studies [BBV09; LZZ11; DB12] agree that a lot of energy is wasted in the tail time of the cellular interface, i.e., the time in which the interface stays in high power mode after a data transfer is completed. Consequently, a number of different approaches were proposed that suggest to adapt this tail time to the actual traffic pattern of the smartphone [QWG+10a; QWG+10b; DB12]. Although, this can help to reduce the tail time and therefore save energy, they rely on the assumption that the tail time is adaptable, which is typically not possible since its length is hard-coded in the device hardware.

Instead of adapting the tail time, several approaches reschedule outgoing traffic to reduce the energy that the interface consumes in the tail time. Liu et al. [LZZ11] propose *TailTheft* that reschedules delay tolerant messages to the tail time of other transmissions. To this end, the user has to provide a delay tolerance threshold for its application via an API. If another transmission occurs before this threshold, the message is sent right after this transmission. Balasubramanian et al. [BBV09] chose a similar approach and additionally propose the idea of prefetching of future content during the tail time of other transmissions. In contrast to the approach presented in this chapter, these approaches propose to send a given message with the next transmission on the device. However, regarding our scenario, position updates should only be sent if the quality constraint of the update protocol is going to be violated. The basic idea of these approaches is implemented by the *nopred* approach that was evaluated in Section 4.6. We have seen that the presented opportunistic update approach is more energy efficient since the number of position updates is less.

An alternative method to decrease energy consumption is by utilizing the different radio interfaces that are available on a mobile device. Given that they differ in characteristics like availability, range, and data rate, one can select the interface for transmission that currently promises the least energy costs. For instance, Pering et al. [PAG+06] propose the *CoolSpots* system that enables a device to save energy by automatically switching between WiFi and Bluetooth. Similarly, Ra et al. [RPS+10] propose a communication link selection strategy that chooses the currently cheapest communication interface based on the delay tolerance of the data and the availability of the networks. However, in many situations there is only a cellular network interface available which renders these approaches ineffective.

4.8 Conclusion

In this chapter, opportunistic extensions to existing position update protocols were presented that significantly improve energy efficiency. It was shown that

by taking into account the characteristics of cellular network interfaces, energy efficiency can be significantly improved by sending position updates opportunistically together with other messages. An approach was presented for optimizing the schedule of opportunistic update messages. This optimization is based on a Markov decision process that predicts the arrival of messages and optimizes the cost of opportunistic messages with respect to energy. The evaluations have shown that the energy efficiency of existing position update protocols—namely, time-based, distance-based, and dead reckoning protocols—can be improved by up to 70 % when using the opportunistic extensions on top of these protocols.

5 Efficient Query Distribution

With the update protocol that was introduced in the last chapter, there is a very efficient way to provide position information of mobile devices to the PS server. This chapter now focuses on the question, how this information can be used to increase the energy efficiency of query distribution. We introduce an algorithm that utilizes the available position information to limit the set of query receivers while distributing a sensing query. Instead of sending a query to all devices, as in the basic sensing system, this algorithm identifies a restricted set of devices which only contains those devices that are very likely to capture data for the query. As a result, all devices that do not receive the query save communication energy. For instance, devices that are moving towards the range of a sensing query are promising candidates for capturing data. Hence, they should be included in the set of query receivers. In contrast, devices that are far-off the query's sensing range would most probably not capture data and should be excluded from the receiver set.

Before the technical details of the query distribution algorithm are introduced, Section 5.1 presents the necessary modifications to the basic system in order to implement the subsequently presented concepts. After that, Section 5.2 formalizes the goal of the query distribution by introducing the problem statement. To be able to quantify which devices are likely to capture data for a given query, a probabilistic sensing model is defined in Section 5.3. This model takes the position information that is provided by the position update protocol as input and defines the probability that a mobile device reaches the sensing range before the sensing deadline. Based on this model, Section 5.4 presents an algorithm that selects a minimum set of devices to receive a given query such that the chances of successfully sensing this query are not negatively affected. Subsequently, Sec-

tion 5.5 introduces an adaptive update control mechanism that automatically disables the update protocol on the mobile devices if only very few queries are issued to the PS system. Section 5.6 presents the evaluation results, before Section 5.7 presents related work in the area of efficient query distribution. Finally, Section 5.8 concludes this chapter.

5.1 Modifications to Basic Sensing System

The query distribution starts when the PS system receives a sensing query from a client. Then, the PS server has to decide to which devices it should send this query. As introduced in Chapter 3, the basic sensing system simply broadcasts the query to all mobile devices $m \in M$ in the service area. However, this results in high energy consumption, since every received message consumes additional energy on a device. Even if the size of the query message is small (in fact, the necessary query information can be encoded in a few bytes), the receiving event powers up the communication interface of the device for typically more than 10 s due to the tail-time characteristics of the cellular network interface (see Section 4.2). This is especially critical for PS systems with large geographical coverage, since they have to distribute many sensing queries. In this case, the communication interface of a single device would only rarely have the chance to power down. In a more sophisticated approach, the server filters out devices that should not receive certain queries and, hence, save energy on devices that do not receive the query message.

To implement this approach, the query manager of the basic system that was introduced in Algorithm 1 is modified according to Algorithm 6. With this modification, a query q is distributed to only a limited set of devices that is defined by a special function (line 2), which will be derived in the course of this chapter. On the mobile device, the sensing engine that was introduced in Algorithm 3 is changed according to Algorithm 7. Here, the only modification is the addition of the position update protocol that was introduced in the last chapter (lines 6–8). Note that this modification ensures that the device sends

Algorithm 6 Modifications to query manager for efficient query distribution.

```
1: procedure QueryManager(q)
2:     R ←calcRecipientSet(p)                          ▷ Modified
3:     for all m ∈ R do
4:         sendQuery(m, q)
5:     end for
6:     ...                                             ▷ As in basic system
```

Algorithm 7 Modifications to sensing engine for efficient query distribution.

```
1: procedure SensingEngine
2:     while TRUE do
3:         posInterval ← 1 s
4:         sleep(posInterval)
5:         p ← getPosition()
6:         if checkUpdateCondition() then       ▷ Added to basic system
7:             sendPositionUpdate(p)
8:         end if
9:         ...                                         ▷ As in basic system
```

forced position updates in order to fulfill the quality constraint of the update protocol. To also send energy-efficient opportunistic updates, the device runs the opportunistic update extension (see Algorithm 5) in parallel to Algorithm 7.

At this point, we neither specify which update protocol the device uses (e.g., time-, distance- or DR-based) nor do we fix the update threshold for the respective protocol to a certain value. We will see later on in the evaluation the effects of different protocols and thresholds on the efficiency of query distribution.

5.2 Problem Statement

The problem with the basic sensing system is that it distributes queries also to devices that cannot participate in sensing until the sensing deadline because of their spatial distance to the sensing range. Obviously, this is an unnecessary waste of energy. In order to tackle this problem, the server limits the distribution of the query to only a subset of devices (called the *recipient set* $R \subseteq M$) that

will be probably able to sense the query. However, this selection has to be done wisely, since choosing a limited recipient set also increases the risk that a sensing query is not sensed by any device, since not all devices are aware of the query. Hence, the goal is to find a minimal recipient set that achieves the same sensing result as the basic sensing system, i.e., when all devices are aware of the query. To formalize this idea, let $\varphi(R) \in \{true, false\}$ denote the successful or unsuccessful sensing of a given query q, when q is distributed to all $m \in R$. As introduced in Chapter 2, we consider a query to be successfully sensed, when at least one device can capture sensor data for it before the query deadline is reached. The goal of the efficient query distribution can now be formulated as follows:

$$\begin{aligned} &\underset{R \subseteq M}{\text{minimize}} \quad |R| \\ &\text{subject to} \quad \varphi(R) = true \end{aligned} \tag{5.1}$$

Having found such an R, sending q only to R instead of M avoids the energy for receiving q on all devices in $M \backslash R$. Note that if we would assume that the server has perfect knowledge about the future movement of devices, finding R would be trivial. In fact, the server could just identify the device $m^* \in M$ that reaches the sensing range of q first and set $R = \{m^*\}$. If no device reaches q within the sensing deadline of q, the server would set $R = \emptyset$.

The challenge of this problem lies in the fact that the PS server does not know for certain the future movement of mobile devices. When deciding on R the server has position information from the position update protocol but it does not know where the devices are moving to. Therefore, it cannot know which devices will actually reach the sensing range of q. For this reason, the subsequently presented approach includes all mobile devices in R that are very likely to move into the range of q before the sensing deadline. This reduces communication energy while the sensing semantics ($\varphi(R) = \varphi(M)$) can be preserved.

To be able to identify the devices that are likely to capture data for q, a probabilistic sensing model is introduced in the next section that predicts the sensing success for each device.

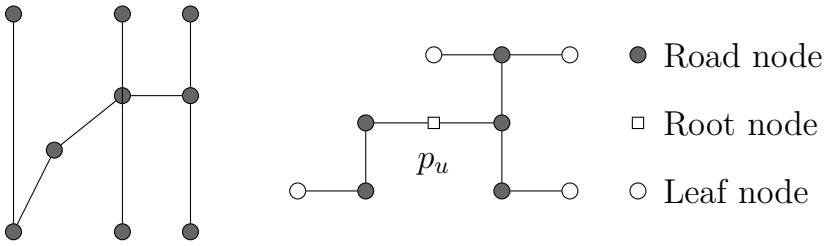
5.3 Probabilistic Sensing Model

This section introduces the probabilistic sensing model in three steps. To compute the sensing probability of a device for given sensing query, it is important to know whether the sensing range of the query lies on the future movement path of the device. Hence, in the first step, the *movement prediction model* is introduced. Based on the position information provided by the position update protocol, this model enumerates all possible movement paths of a device and assigns each path a probability that the device will actually take this path. In the second step, the *single path sensing probability* for a device is defined. Here, we assume that the future path of a device m is known and formulate the probability that the device can read data for a query q when it moves along this specific path. Finally, the *multiple path sensing probability* combines both models in order to define the overall probability that m can read data for q.

5.3.1 Movement Prediction Model

As described in the system model in Section 2.1, mobile devices move on a road graph that is known to the PS server. This road graph is defined by a set of *road nodes* and a set of edges that interconnect these nodes (see Figure 5.1a). For each road node r, the set of its adjacent road nodes is given as A_r. To reflect the significance of different roads, an individual turning function $f_r : A_r \times A_r \rightarrow [0, 1]$ is given for each road node r. The value of $f_r(r_a, r_b)$ denotes the probability that a mobile device leaves road node r towards road node r_b when having entered it via road node r_a. This function can for example be derived from road traffic studies [Mah84], in which movement patterns of pedestrians are analyzed in order to infer statistical information about turning probabilities.

For predicting the movement path of a device m, we enumerate the different paths that the device can take starting from its last known position p_u, which is the position that is provided by the position update protocol. However, since the number of possible paths grows exponentially over time, we limit this number by assuming that a device always moves towards some unknown destination on



(a) Road graph (b) Tree of possible movement

Figure 5.1: Road graph and movement tree.

the graph traversing the shortest path. This assumption follows the intuition that a person wants to arrive at a certain point of interest as fast as possible and therefore chooses the shortest way to reach that point. This is also a basic assumption that is used by many sophisticated mobility models, for instance the random waypoint model [BHP04], or for trajectory matching [ESS+11]. Following this assumption, we can describe the movement of a device m between time t_u (the time of the last position update p_u) and the query sensing deadline t_d by a tree of road nodes which is rooted at *root node* p_u (see Figure 5.1b). The *leaf nodes* of this tree are the set of all points on the road graph that are distance $s_{max} = (t_d - t_u) \cdot v_{max}$ away from p_u. Here, v_{max} depicts the maximum speed of device m. This value can either be obtained directly from the device or some value about the maximum device speed can be assumed (for instance, by taking into account speed limits that are included in the road graph). Each path from the root node towards a leaf describes one possible movement path w that m can take until t_d. We describe such a path w by the sequence of nodes that are traversed on this path, i.e., $w = (r_{root}, \cdots, r_{leaf})$.

To model a probability distribution over all possible paths, we introduce the random variable X_w that reflects the future movement path of a device. The probability that a device m takes a certain path w is then given as $P(X_w = w)$. To derive the probability distribution over X_w, we multiply the respective turning probabilities at the encountered road nodes when traversing a certain

path w:

$$P(X_w = w) = \prod_{i=1}^{i<|w|-1} f_{w[i]}(w[i-1], w[i+1]) \tag{5.2}$$

As a result, we get for each movement path of a device a probability value. In the next step, we define the probability that a device can read data for a given query under the condition that we know the path on which the device is moving.

5.3.2 Single Path Sensing Probability

To denote the sensing success of a single device, we introduce the random variable $X_s \in \{0, 1\}$, which reflects whether a device m can read data for a given query q (denoted as $X_s = 1$) or not ($X_s = 0$). In this section, the probability distribution over X_s is derived under the assumption that m takes a certain path w, i.e., $P(X_s = 1 \mid X_w = w)$. Using this definition and the probabilistic information provided by the movement prediction model, the general distribution $P(X_s = 1)$ is then defined in the next section. In the following, we will use the short term $P(X_s)$ instead of $P(X_s = 1)$ and refer to this probability as *sensing probability*.

As already mentioned, we assume that m traverses path w after starting from point p_u. In order to be able to read data for query q when following path w, the sensing range of q must intersect with w, otherwise the sensing probability for this path is $P(X_s \mid X_w = w) = 0$. In case path w intersects with the sensing range, the enclosed line segment $[p_{i_1}, p_{i_2}]$ contains the points from which m can read data for q (see Figure 5.2).

The existence of such an intersection ensures that the *spatial* condition for sensing is fulfilled. However, to determine whether m can read data for q, we also have to check the *temporal* condition. More precisely, m needs to pass line segment $[p_{i_1}, p_{i_2}]$ in the time frame between the time when q is queried (referred to as t_q) and the sensing deadline t_d of the query. To this end, we derive in the following a probabilistic representation of the position of device m between time t_q and t_d. Based on this, we can derive the likelihood that the temporal condition for sensing is fulfilled.

To consider a device's moving speed, we assume that a probability density

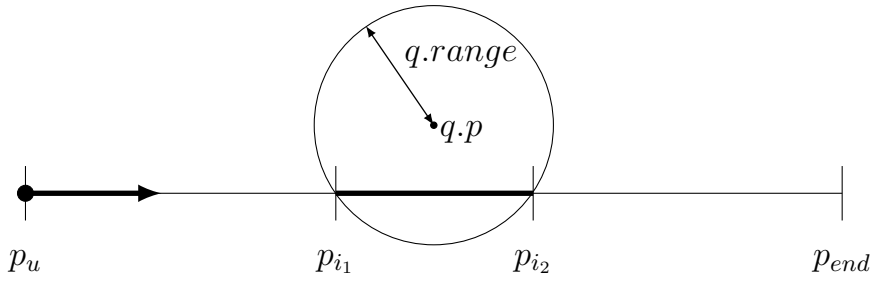


Figure 5.2: Movement path of a device including the line segment $[p_{i_1}, p_{i_2}]$ from which the device can read data for query q.

function (pdf) $f_{X_v}(v)$ of the random variable X_v is given, which describes the average speed of the device for an arbitrary time interval. Furthermore, we assume that the values of X_v for two disjoint time intervals are independent, since we assume that a device can change its speed independently in any time interval. This pdf can either be provided directly by the device or we can assume an appropriate probability distribution to approximate the device speed (e.g., a Gaussian distribution). The distance d that a device passes in a given time period Δt is then given by the following formula:

$$d = X_v \cdot \Delta t$$

We denote the distance that m passes on w between time t_u (when the last location update from the device was received) and the time t_q (when the query was issued) as X_{d_1}. Note that this is the distance that the device has already passed since the last location update. Analogously, we denote the distance that m passes between t_q and t_d as X_{d_2}, which is the distance that the device will pass until the sensing deadline. Both are defined as follows:

$$X_{d_1} = X_v \cdot (t_q - t_u)$$
$$X_{d_2} = X_v \cdot (t_d - t_q)$$

The two variables X_{d_1} and X_{d_2} are a linear transformation of the random variable X_v. Hence, they are both random variables as well. The pdfs of these linearly

transformed variables are then given as:

$$f_{X_{d_1}}(d_1) = \frac{1}{t_q - t_u} \cdot f_{X_v}\left(\frac{d_1}{t_q - t_u}\right)$$
$$f_{X_{d_2}}(d_2) = \frac{1}{t_d - t_q} \cdot f_{X_v}\left(\frac{d_2}{t_d - t_q}\right)$$

Based on these probabilistic definitions, the position of m on path w at time t_q is given as $p_u + X_{d_1}$ and as $p_u + X_{d_1} + X_{d_2}$ for time t_d. We can now formulate the case in which m passes line segment $[p_{i_1}, p_{i_2}]$ in the time frame $[t_q, t_d]$ by the following two conditions:

1. When m receives q from the server at time t_q, it must not have moved further than point p_{i_2} on w. Otherwise m has already passed q when it receives the query.

2. When the sensing deadline of q is reached at time t_d, m must have reached at least point p_{i_1} on w.

More formally, we say a device m can read data for q on a given path if the following two conditions are fulfilled:

$$X_{d_1} \leq \overline{p_{i_2}} \tag{5.3}$$
$$X_{d_1} + X_{d_2} \geq \overline{p_{i_1}} \tag{5.4}$$

Here, $\overline{p_i}$ depicts the distance between p_u and p_i on path w. Note that due to our shortest path assumption, we can assume that a device only moves towards the end of path w.

To formulate these two conditions in a closed form, we need to define the joint pdf of the random variables X_{d_1} and $X_{d_1} + X_{d_2}$. Due to the independence of X_{d_1} and X_{d_2}, we get the convolution of the pdf of both variables as:

$$\begin{aligned} f_{X_{d_1}, X_{d_1}+X_{d_2}}(d_1, d) &= f_{X_{d_1}, X_{d_2}}(d_1, d - d_1) \\ &= f_{X_{d_1}}(d_1) \cdot f_{X_{d_2}}(d - d_1) \end{aligned}$$

We can now formulate the sensing probability for q when m moves on path w by integrating this joint pdf over the boundaries that we defined in Equation 5.3 and 5.4:

$$\begin{aligned} & P(X_s \mid X_w = w) \\ &= P(X_{d_1} \leq p_{i_2}, p_{i_1} \leq X_{d_1} + X_{d_2} \leq p_{end}) \\ &= \int_0^{\overline{p_{i_2}}} \int_{\overline{p_{i_1}}}^{\overline{p_{end}}} f_{X_{d_1}}(x) \cdot f_{X_{d_2}}(y - x)\, dydx \end{aligned} \tag{5.5}$$

Here, p_{end} refers to the end of path w (see Figure 5.2).

Note that for the calculation of the sensing probability we assumed that there are only two intersection points between w and the sensing range of q. Cases in which the sensing range intersects path w on more than two points can be handled by summing up the single sensing probabilities for each intersection using the inclusion-exclusion method [Cam94].

5.3.3 Multiple Path Sensing Probability

To get the overall sensing probability $P(X_s)$, which is independent from a specific path, we combine the definitions from Equations 5.2 and 5.5. Given the probability that device m moves on path w and the probability that m can read data for q if it moves on w, we apply Bayes' Theorem as follows:

$$P(X_s) = \sum_{\forall w_i} P(X_s \mid w_i) \cdot P(X_w = w_i) \tag{5.6}$$

In the following, we will denote the sensing probability $P(X_s)$ for a certain device m and a given query q as $p_{m,q}$.

5.4 Recipient Selection Algorithm

We can now use the introduced definitions to find a recipient set for the query distribution. The idea is to first introduce the *joint sensing probability* for a set of devices and then use this metric to find a recipient set that satisfies the quality constraints that is required by the client that issued the sensing query.

To quantify this quality, we use the quality parameter σ that is included in the definition of a sensing query, as introduced in Section 2.2.

Assuming that q is distributed to all $m \in D$, the joint sensing probability for a given set of devices D is defined as the probability that query q can be read by at least one device (which is sufficient for successfully sensing the query). To this end, we use the sensing probability $p_{m,q}$ for a single device from Equation 5.6 and calculate the joint sensing probability p_q for D as:

$$p_q(D) = 1 - \prod_{\forall m \in D} (1 - p_{m,q}) \tag{5.7}$$

This function can be derived from the special case of a binomial distribution, in which at least one event is true. Note that we could easily extend the earlier defined query semantics by requiring that a sensing query is only satisfied if k different devices read data for it. This can for instance be necessary if a client requires more than one sensor value from the query range. In this case, Equation 5.7 can be extended to consider k events (see [Cam94] for the combinatorial details), while all other parts of the system stay untouched.

To find a promising recipient set R, we can use the above definition to quantify the sensing success of different $R \in 2^M$. Obviously, $p_q(R)$ would be maximized if we set $R = M$, i.e., including all devices in R. However, since our goal is to increase energy efficiency, we try to find a small R that still promises high sensing success $p_q(R)$. Finding such an R requires that the sensing probability $p_{m,q}$ is known for all $m \in M$. Considering the computational effort for calculating $p_{m,q}$, this is hardly scalable if there is a very large number of mobile devices in the PS system. Hence, we first limit the solution space by restricting the number of devices that are candidates for being included in R.

As already mentioned, only mobile devices can read data that can reach the sensing range of q before its sensing deadline. We can use this property to apply a filter on the set of mobile devices, which excludes all devices that are too far away to reach q in time. To implement this filtering operation in an efficient way, we only check if the Euclidean distance between the last known position

p_u^m of device m and the sensing range of q is smaller than the maximum distance that m can travel before the sensing deadline. For this purpose, we introduce the following predicate that maps to this condition:

$$\phi(m) : d_{Eucl}(p_u^m, q.p - q.range) < v_{max}^m \cdot (t_d - t_u)$$

We can now define the *candidate set* C for the recipient set selection as:

$$C = \{m \in M \mid \phi(m)\}$$

Having defined this set, $p_{m,q}$ is subsequently calculated for only those devices that are included in C.

The goal is now to include as many devices in R such that the joint sensing probability of these devices reaches the client defined quality parameter $\sigma \in (0, 1]$. In case σ cannot be reached (since not enough devices have high sensing probability), a best effort solution should be provided that maximizes the joint sensing probability. To implement this goal, we define R by greedily including the mobile device with the highest sensing probability in C and use σ to restrict the number of devices that will be included into R. More precisely, we start by setting $R = \emptyset$ and then include devices in R until $p_q(R)$ reaches the value of σ. As a result, the higher σ is set, the more devices are included in R. Algorithm 8 shows the calculation of R in detail. The function called in line 2 calculates the candidate set out of all devices, as shown before, while the function in line 6 returns the device with the currently largest value of $p_{m,q}$ in C. The algorithm stops when the joint sensing probability of R reaches the quality parameter σ or when R is equivalent to C. In the latter case, quality parameter σ cannot be reached. However, to provide a best effort solution by maximizing the sensing probability the algorithm returns R, which is then equal to the candidate set C.

Algorithm 8 Calculation of the recipient set R.

```
1: procedure calcRecipientSet(q)
2:     C ← filterCandidates(R, q)
3:     R ← ∅
4:     p_q ← 1
5:     repeat
6:         m ← getMaxProbDevice(C)
7:         R ← R ∪ {m}
8:         C ← C \ {m}
9:         p_q ← p_q · (1 − p_{m,q})
10:    until (1 − p_q ≥ q.σ) ∨ (R = C)
11:    return R
12: end procedure
```

5.5 Adaptive Update Control

The presented query distribution is especially energy efficient if the PS system has to handle a high query load, i.e., when a large number of sensing queries are issued to the system. However, if queries are issued only rarely, the efficiency of this approach decreases. If no queries are issued at all, it is even more energy consuming than the basic sensing system, since devices still run the position update protocol that consumes energy. To account for this fact, this section introduces a concept that temporarily disables the update protocol on a device if it moves in an area in which the query load is low.

To reflect the spatial variations of the query load, a grid is introduced that covers the service area. The size of a grid cell is a system parameter and we assume it to be in the magnitude of several hundred meters, since a small grid cell sizes implies communication overhead on the devices, as we will see next. For each grid cell c, the PS server keeps track of the number of queries which are located within that cell and then computes the average interarrival time λ_c between two queries. If λ_c is bigger than some predefined threshold λ_{min} the state of cell c is set to *passive*, otherwise it is set to *active*. These cell state values are initially distributed to the device and updated whenever the state of

a cell changes. To distribute them efficiently, they can be encoded as a binary string and attached when sending sensing queries to a device. With the help of this information, a device activates and deactivates its position update protocol according to the status of the cell in which it is currently located.

In general, we assume that the query load for one cell is constant over time. In fact, there are places that are especially interesting for placing queries, e.g., city centers, and places that are not very busy, thus, queries are issued only rarely, e.g., in suburban areas. As a result, the PS server rarely has to update the status values of the cells on the devices. In this thesis, we only consider this basic version of the adaptive update control. However, to also address temporal variations in the query load, the server could use a more sophisticated query arrival model to predict the query load. For instance, the server could provide the devices with a predictor, which can be used by the devices to predict the current query load using their current location and time as input. Such a predictor can be learned by the server over time using advanced prediction techniques like time series analysis or artificial neural networks [Cho03].

5.6 Evaluation

In this section, we look at the evaluation results of the efficient query distribution. Before the results are presented, the simulation setup and the energy model are introduced.

5.6.1 Simulation Setup

The concepts presented in this section were implemented with the Opportunistic Network Environment (ONE) simulator [KOK09], since real-world experiments would require a large number of physical devices and participants. Therefore, they are only hardly feasible. The ONE simulator is a Java based mobility simulator that is capable of generating realistic node movement by providing map-based movement models which constrain the node movement to predetermined paths [KOK09]. For simulating device mobility, the included mobility trace files

of the simulator were used, which realize a random shortest path movement of pedestrians on the road graph of Helsinki that has a size of $15\,\mathrm{km}^2$. This road graph is also considered as the service area of the PS system. We assume to not have any prior knowledge about the turning probabilities at intersections on this map. Thus, we assume that each road at an intersection is equally likely to be taken by a device. To allow for the calculation of sensing probabilities for several hundreds of devices in real-time on the PS server, the integrals for the calculation of the probabilistic sensing model were numerically approximated. The simulator generated three hours of pedestrian movement on the given road graph. In the evaluation the following approaches are compared:

basic The basic sensing system that broadcasts a sensing query to all mobile devices. Using this approach, the devices do not run any kind of position update protocol.

dist The efficient query distribution in which devices use a distance-based update protocol to update their positions on the PS server.

dr The efficient query distribution in which devices use a dead reckoning update protocol to update their positions on the PS server.

For the following results, different parameters of the PS system were varied. While one specific parameter was varied for one analysis, the other parameters stayed fixed during that experiment. For this purpose, the following standard parameter configuration was chosen: One sensing query per minute was generated at a random position on the map, which expires after a the sensing deadline of 300 s. The accuracy of the update protocol that is used in the *dist-* and *dr-*approach is set to 200 m. The number of mobile devices in the PS system is set to 900. Furthermore, the quality parameter σ of a sensing query is set to $\sigma = 1.0$. We will use a the notation $approach - \sigma$ to denote the case when σ is different from 1.0. For instance, *dr-0.8* means that the *dr*-approach is used with $\sigma = 0.8$.

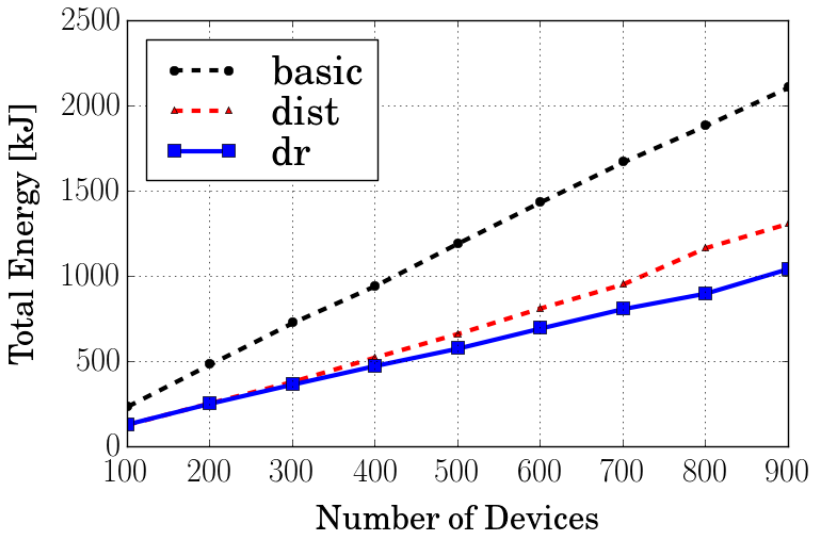


Figure 5.3: Energy consumption for different number of devices in the service area.

5.6.2 Energy Model

For evaluating the energy consumption of the approaches, we only consider the energy that is needed on the device for communication, i.e., for receiving sensing queries and sending position updates. To measure this communication energy, the energy model is used that was introduced for evaluating the position update protocol in Section 4.2. Also considering the energy that is consumed for positioning would add the same amount of energy in all approaches (since all approaches use continuous positioning) and is therefore not decisive.

5.6.3 Energy Efficiency

First, we look at the energy efficiency of the query distribution. For this purpose, we sum up the energy that each device consumes to get the *total energy* that is consumed by all devices together. Figure 5.3 shows how the energy consumption varies for different number of devices that are present in the service area. Over all scenarios, the optimization with the dead reckoning update protocol is the most energy-efficient approach. We will investigate in the following subsections

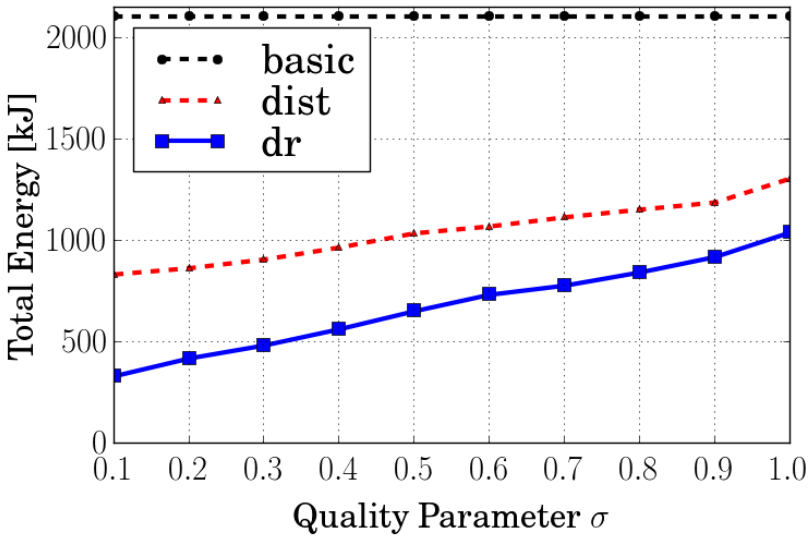


Figure 5.4: Energy consumption for different values of the quality parameter σ.

in detail why it is the most efficient approach. It saves on average 50 % of energy compared to the basic approach, while the distance approach saves on average 43 % with respect to the basic approach. Obviously, the overall energy consumption increases with more devices in the PS system. However, we can also see that the energy savings of the distance and DR-approach increase with more number of devices in the service area.

Next, in Figure 5.4 we can see how the quality parameter σ influences the energy consumption of the system. While the energy consumption of the basic approach is not affected by this parameter, the energy of the optimized approaches decreases with smaller values of σ. That reflects the intended semantics of parameter σ, which can be specified by the client in order to control the amount of system resources that should be utilized to answer its query. These results can be explained by the fact that parameter σ controls the number of devices that are included in the recipient set. For instance, a small value of σ results in a small recipient set and leads to more energy savings.

To see how the accuracy parameter δ of the underlying position update protocol influences the energy consumption of the system, we look in Figure 5.5 at different values of δ_D, respectively δ_{DR}. In general, smaller values of δ result

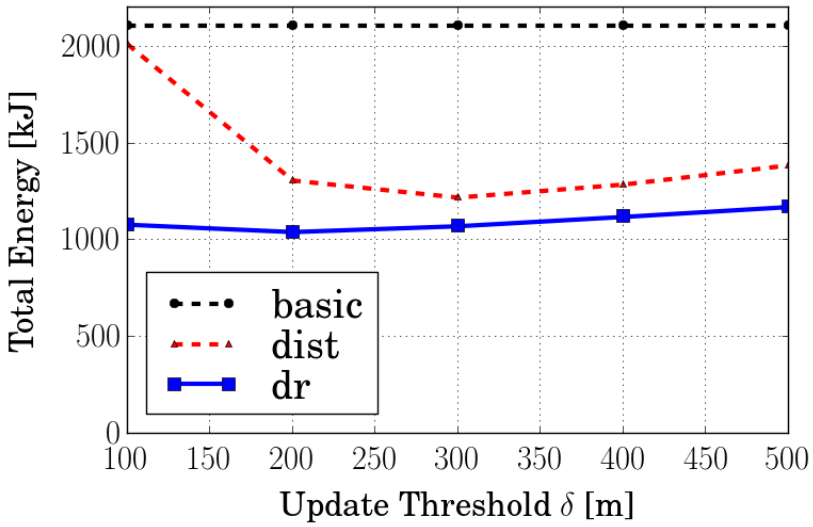


Figure 5.5: Energy consumption for different accuracy parameters of the underlying position update protocol.

in more position updates that are sent by a device to the server. As a result, the device position on the server is more accurate. However, more updates also result in more energy overhead for sending them. We see from the figure that for both approaches, there is a scenario in which the energy consumption is minimal, i.e., when $\delta_D = 300\,\text{m}$ and $\delta_{DR} = 200\,\text{m}$. The existence of such a minimum can be explained as follows. If δ is smaller than this optimal value, the accuracy of the update protocol requires the sending of many position updates and results in a high energy consumption for sending them. In contrast, if δ is larger than the optimal value, the device position on the server is not accurate enough for the query distribution to work efficiently. As a result, the query distribution cannot identify a small recipient set and too many devices receive a query, i.e., the energy consumption is high again.

Finally, we look at the efficiency of the PS system for different numbers of queries that are requested by the clients. This is implemented by varying the interarrival times between two queries that are issued to the system, i.e., larger interarrival times lead to less queries that are issued to the system. Figure 5.6 shows the energy consumption without using the adaptive update control that

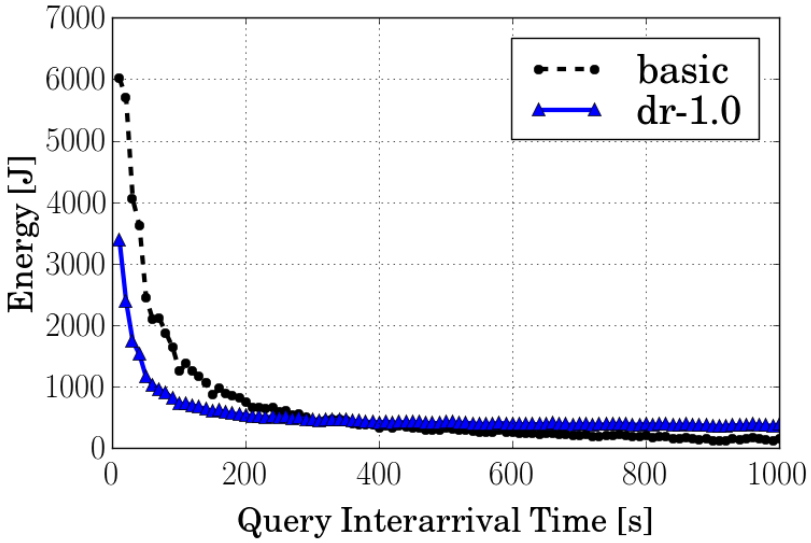


Figure 5.6: Energy consumption for different query interarrival times without the adaptive update control.

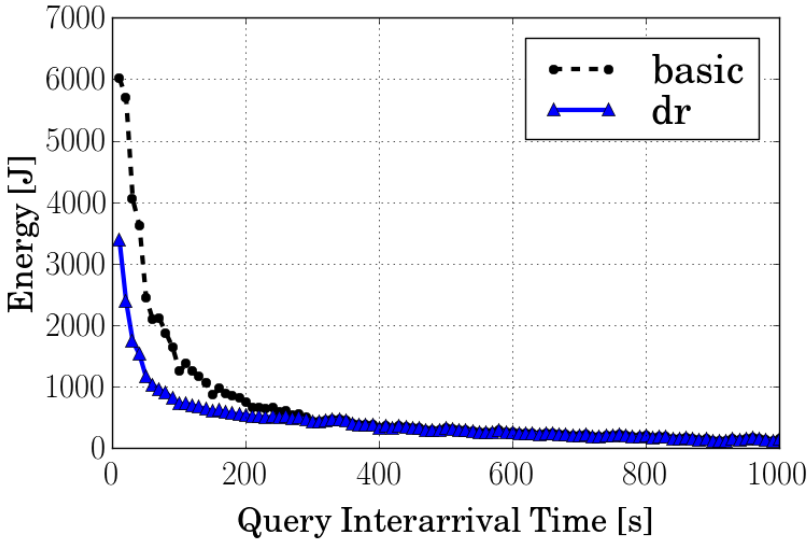


Figure 5.7: Energy consumption for different query interarrival times using the adaptive update control.

was introduced in Section 5.5. We see that the optimized approaches perform better than the basic approach when queries are frequently issued. On the other hand, a larger query interarrival time leads to less energy savings of the optimized approach. If the query interarrival time is larger than 350 s, the basic approach is even more energy efficient. In this case, the energy overhead for running the position update protocol is too big and cannot be compensated by the more efficient query distribution. This figure clearly shows that the proposed approach can save a lot of energy if the query load in the PS systems is high, but also indicates that the PS system should fall back to the basic approach if only few queries are present. To adaptively switch between these strategies, Figure 5.7 shows the same scenario when using the adaptive update control. For this purpose, the whole service area is considered as one grid cell and the threshold for activating the update protocol is set to $\lambda_{min} = 350\,\text{s}$ seconds. We see that for small query interarrival times the optimized approach performs better and for large interarrival times the energy consumption for both approaches are equivalent. Hence, the system automatically falls back to the basic approach if the query load is low.

5.6.4 Sensing Effectiveness

To see if the shown energy reduction has a negative impact on the sensing effectiveness, we compare the number of successfully satisfied queries. First, we compare this number for different number of devices in Figure 5.8. Obviously, with more devices in a scenario more sensing queries can be satisfied. Moreover, we see that the optimized approach that uses the DR protocol with $\sigma = 1.0$, satisfies the same number of queries as the basic approach. If σ decreases, the number of satisfied queries decreases accordingly. To investigate this in more detail, Figure 5.9 shows the number of satisfied queries for all values of σ. The distance approach reaches for $\sigma = 0.1$ a relative sensing success of 72 % which goes up to 100 % when $\sigma \geq 0.7$. In comparison, the dead reckoning approach has a much lower sensing success for smaller values of σ, i.e, for $\sigma = 0.1$ it reaches 37 % which also goes up to 100 % for higher values of σ. One other thing

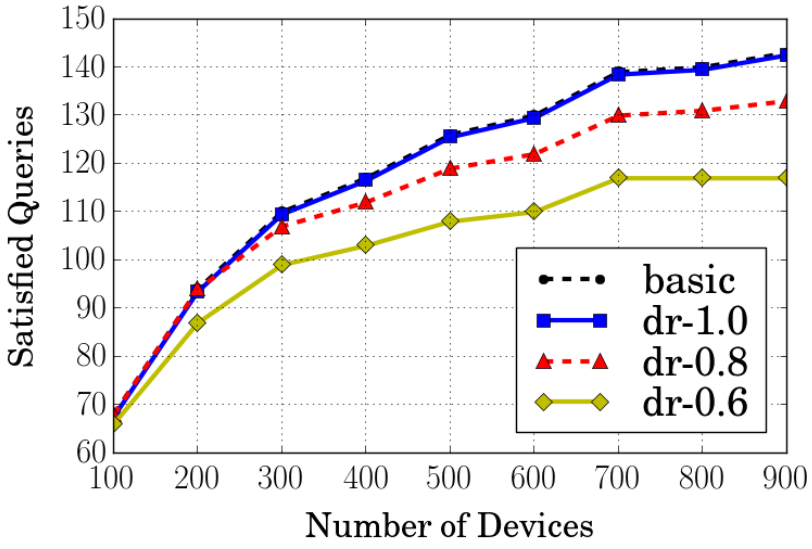


Figure 5.8: Sensing effectiveness for different number of devices in the service area.

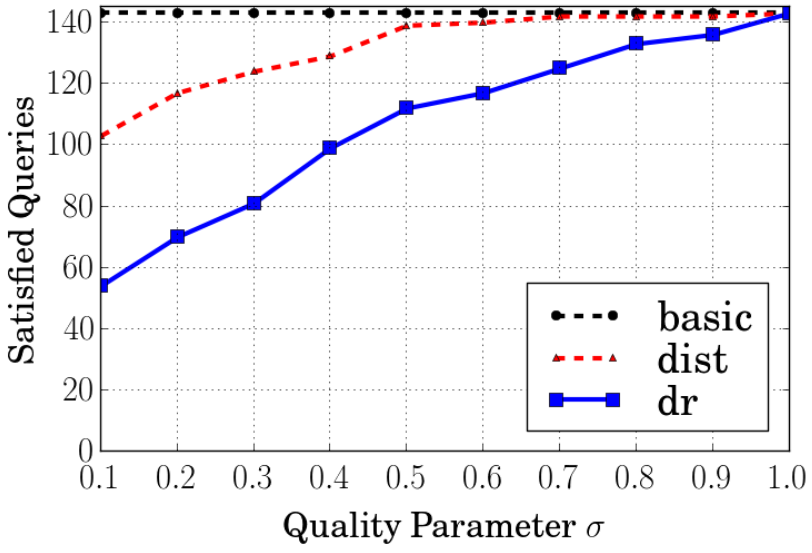


Figure 5.9: Sensing effectiveness for different values of quality parameter σ.

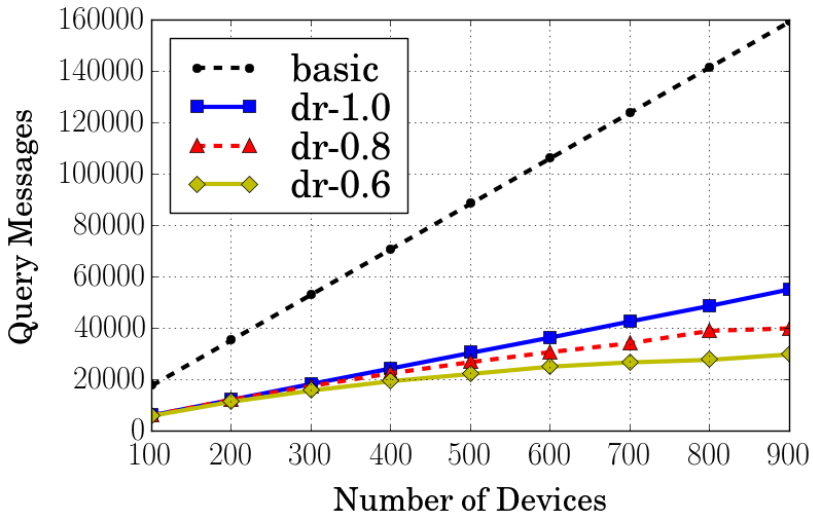


Figure 5.10: Message overhead for sending sensing queries.

that we can see from Figure 5.9 is that already a small value of σ is sufficient to successfully sense the majority of queries. Overall, the results show that a reduced value of quality parameter σ not only decreases the energy consumption of the system, but also reduces the sensing effectiveness.

5.6.5 Message Overhead

To understand how the presented energy values are composed, we look at the different messages that are sent in the system. Figure 5.10 shows the message overhead for sending sensing queries. We see that the basic approach sends a large number of query messages, while only a fraction of these messages are sent for the optimized approach. In comparison, Figure 5.11 shows the number of update messages that are sent. As we have seen already in the last chapter, the dead reckoning protocol only needs a fraction of messages compared to the distance-based approach. In contrast, the basic approach does not require any location update messages. However, this advantage is compensated by the high number of sensing queries that are distributed.

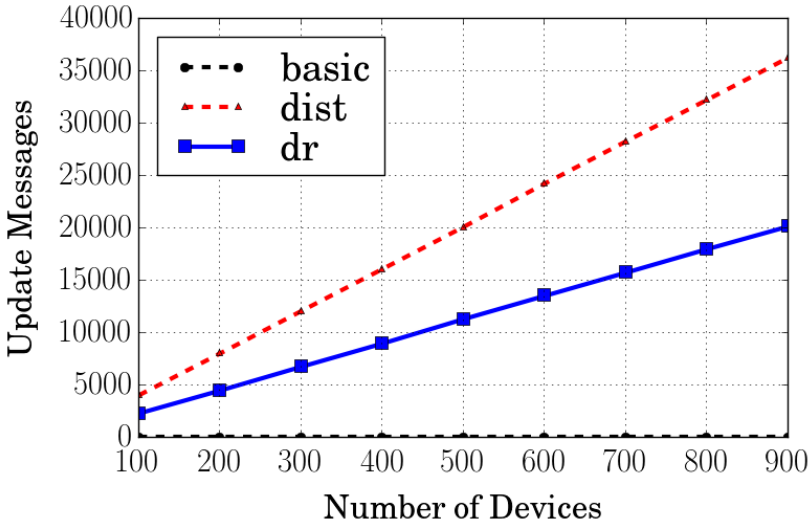


Figure 5.11: Message overhead for sending position updates.

5.6.6 Summary

The results and findings of this evaluation can be summarized as follows. The efficient query distribution significantly lowers the energy consumption of devices in contrast to the basic approach, because of the reduced amount of query messages a device receives from the PS server. However, this approach requires that each device runs a position update protocol to report its positions to the PS server. The measurements showed that the energy overhead for this protocol is less than the savings that can be achieved by the query distribution. Furthermore, we saw that this approach is especially beneficial in cases when many queries are distributed by the PS server.

5.7 Related Work

Before we conclude this chapter, we have a look on related work in the field of efficient query distribution. However, only a few approaches have been proposed so far in this area. Lu et al. [LLE+10] showed with their "Bubble-Sense" system how a sensing query can be "pinned" to a physical location by periodically exchanging ad-hoc messages between the devices near that location. However,

this concept relies on the wide availability of ad-hoc communication interfaces on all devices, which are still hard to configure and therefore are rarely available in practice. Moreover, if no device is nearby the location of the sensing query, the query is lost and can only be restored by a node that has stored the query.

Other concepts that are more closely related were proposed by Reddy et al. [RES10], Ruan et al. [RNL11] and Cardone et al. [CFB+13]. Similar to the presented approach, their systems choose a subset of mobile devices to which a sensing query is distributed. To decide which devices are most suitable for being included into that subset, their approaches analyze historic information about the users movement patterns and extract places were the user regularly resides. However, this requires that the user agrees to store a privacy critical user profile of its movement history. Moreover, these approaches do not consider the energy that is required for generating these profiles and keeping them up-to-date on the server. The query distribution approach that was presented in this chapter uses a different strategy. More precisely, it does not analyze user profiles but rather estimates the future movement of a user based only on its last known position.

Another related approach was presented by Hachem et al. [HPI13], which propose a probabilistic device registration mechanism. Rather than distributing sensing queries to devices, a mobile device registers at the PS server if it is willing to receive queries for the current region in which it moves. The introduced probabilistic registration approach reduces the number of device registrations by lowering the probability for a device to register if there are several other devices on the same movement path. In contrast to the presented approach, they assume that a device knows its future movement path, while the prediction of the movement of other devices is solely based on a spatial distribution model.

Finally the PRISM platform [DMP+10] follows the same idea as the presented approach by proactively updating device resources on the PS server to enhance query distribution. With the help of the resource information, PRISM automatically pushes queries to an appropriate set of devices. However, the system does not give details on the device selection and proposes the use of customizable predicates to filter out devices that are not appropriate to receive a given query.

5.8 Conclusion

This chapter introduced an algorithm that efficiently distributes sensing queries from the PS server to mobile devices. In comparison to the basic sensing system that simply broadcasts sensing queries, this approach distributes queries selectively by utilizing the information provided by the position update protocol. A probabilistic sensing model was presented that is used to restrict the distribution of a query to those devices that are best suited for capturing sensor data for that query. The evaluation showed that this approach can significantly increase the energy efficiency of query distribution, especially when the query load of the PS system is very high.

6 Efficient Outdoor Position Sensing

So far, we have assumed that a mobile device is always aware of its current position and did not consider the energy for sensing its position. Considering the proliferation of location-based services, the availability of the current device position is a valid assumption, since applications like friend finders or navigation systems typically require accurate position information. In outdoor scenarios, these applications require a device to continuously sample its GPS sensor. As a result, the device position is also available to the PS application without any extra energy overhead.

From this chapter on, we drop the assumption that the PS system can access the current device position without any additional overhead. Instead, we consider the case in which no other application on the device requires position information. Consequently, the energy that is consumed to sense a device's position is fully accounted to the PS system. To avoid confusion with the energy that is consumed for sensing environmental data, we refer in the following to the energy that is needed for position sensing as *positioning energy*.

The PS system that was used so far relies on continuous positioning, since a device needs the current position to check if a sensing range is reached and to trigger position updates. However, in today's mobile devices GPS positioning is still one of the most energy consuming operations [LYL+10], which can reduce the battery lifetime of a device significantly. As a result, users typically avoid running applications that need continuous position information since they are aware of the high energy overhead for positioning. To tackle this drawback, this chapter introduces a modification to the PS system that avoids continuous positioning on the device and, hence, reduces positioning energy. Note that in contrast to the position update protocol that was introduced in Chapter 4,

this approach does not tackle the sending of position updates to the PS server. Instead it provides algorithms that modify the underlying positioning system, which provides the position information for the efficient update protocol.

Existing work on energy-efficient position sensing [KLG+09; FRC11] has shown that one possibility to reduce positioning energy is the adaption of the sampling rate of the GPS sensor. However, finding an efficient strategy for sampling the GPS sensor in a PS scenario is not straightforward. Simply reducing the interval between two consecutive position fixes would have an undesired impact on the sensing result. For instance, a device may miss to read data for a sensing query while passing through the range of that query as the GPS sensor is temporarily deactivated. Hence, the approach presented in this chapter controls the operation of the GPS sensor in such a way that the energy for positioning is significantly reduced while the quantity of captured sensor data does not decline. This is achieved by adapting the time until a mobile device performs its next position fix to the distance to the nearest sensing range. For instance, considering the case that a noise level measurement is requested at some location l in the city. Using the basic sensing system, a mobile device m would continuously fix its position to determine whether it is close enough for recording a sound sample at l, even when it is still far away from l. In contrast, the approach that is presented in this chapter determines an optimal time span by which m can delay the next position fix until m is close enough for recording at l.

This rest of this chapter is organized as follows: In order to find an energy-efficient position sensing algorithm, we need to understand the energy characteristics of the GPS sensor, which are introduced in Section 6.1. Subsequently, the adaptive positioning scheme is introduced in Section 6.2, which temporarily deactivates the GPS sensor to save energy. The use of adaptive positioning has implications on other parts of the sensing system, namely the query listener and the position update protocol, which both run on the devices. Hence, Section 6.3 and Section 6.4 introduce modifications to these components, before Section 6.5 presents the evaluation results of this approach. Finally, Section 6.6 summarizes related work on this topic before Section 6.7 concludes this chapter.

6.1 GPS Energy Model

To understand the concepts that are introduced in this chapter, we start by looking at the energy characteristics of a GPS sensor. In general, the energy consumption of a GPS sensor is non-linear, i.e., the consumed energy is not a constant factor of the position fixes that were performed. This is attributed to the fact that such a sensor has a certain request delay to switch from idle state to the operational state. Additionally, once powered up, the sensor does not immediately power down after a position fix is completed. Instead, it stays in the operational state for a certain time frame to avoid the request delay for subsequent position requests. As a result, the total energy consumption not only depends on the number of position fixes but also on the time intervals between them. This behavior is very similar to what we have seen before when looking at the energy characteristics of a cellular communication interface of a mobile device (see Section 4.2).

To investigate these characteristics in more detail, we have a look at the measurement results that were published by Kjærgaard et al. [KLG+09]. In general, a GPS sensor has two power states: *IDLE* and *ON*. According to the measurements, a mobile device consumes in the *IDLE* state on average 0.0621 W, while it consumes 0.324 W on average in the *ON* state. To perform a position fix, the sensor has to be in the *ON* state. After a position fix is completed, the sensor stays for another 30 s in the *ON* state, before it switches back to the *IDLE* state. This time frame is called the *power-off delay*. Once the sensor switched back to the *IDLE* state, it needs some time to power up again and is then able to provide a new position after 6 s. This time frame is called the *request delay*. Table 6.1 gives an overview of all introduced values.

To illustrate these energy characteristics, Figure 6.1 shows an example that includes the different power states and delays. At the beginning of the example (at $t = 0\,\text{s}$) the GPS sensor is in *IDLE* mode. At $t = 10\,\text{s}$, some application requests a position from the sensor. Hence, the sensor switches to the *ON* state and starts calculating the current device position. At $t = 16\,\text{s}$, this calculation is

Table 6.1: Energy characteristics of a GPS sensor.

Operating State	Power [W]
IDLE	0.0621
ON	0.324

Symbol	Delay Type	Delay [s]
$delay_{req}$	Request delay	6
$delay_{off}$	Power-off delay	30

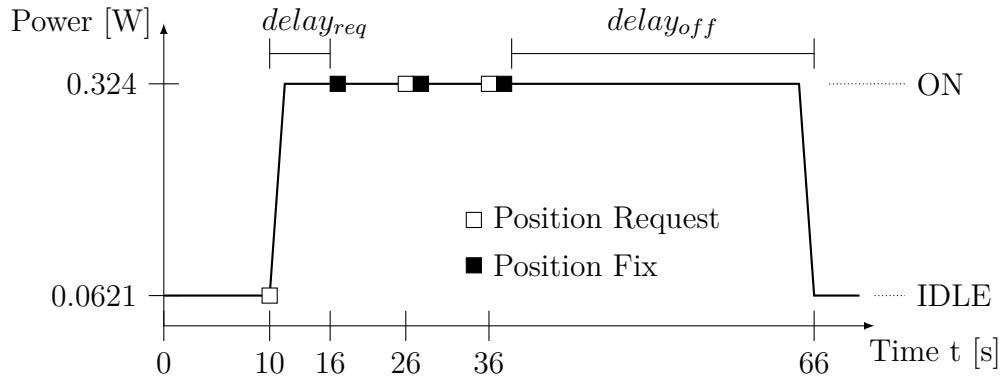


Figure 6.1: Example of the energy consumption of a GPS sensor over time.

finished and the position is provided back to the application. If no other position request would arrive, the sensor would power down to the *IDLE* state after the lapse of the power-off delay. However, at $t = 26\,\text{s}$ and $t = 36\,\text{s}$ subsequent position requests arrive at the sensor. Since the sensor is already in the *ON* state, it can serve these requests immediately. After the last position fix at time $t = 36\,\text{s}$, the sensor stays for the length of the power-off delay in the *ON* state until it powers down at $t = 66\,\text{s}$.

We can see from this example, that the different power states and delays of the GPS sensor need to be considered when designing energy-efficient position sensing algorithms. For instance, avoiding continuous positioning by simply increasing the delay between two position requests to 10 seconds is not an efficient strategy. In this case, the sensor would still consume as much energy as for continuous positioning, since it cannot power down between two requests.

6.2 Adaptive Positioning

The concept of continuous positioning ensures that no mobile device passes the range of a sensing query without being aware of it. Thus, this approach can be used as a reference for achieving optimal sensing effectiveness. Nevertheless, this approach also causes high energy consumption since mobile devices continuously fix their position. To tackle this problem this section introduces the adaptive positioning approach, which suppresses position fixes if the range of the closest sensing query is some distance away.

To introduce this approach, we first look on the modifications that need to be done on the device in order to implement adaptive positioning. Subsequently, the concept of adaptive positioning is introduced.

6.2.1 Modification to Basic System

To implement adaptive positioning, the sensing engine of the basic sensing system needs to be modified according to Algorithm 9. Compared to the non-modified algorithm, the length of the *posInterval* in line 4 is no longer fixed to a fixed time interval c_p but is now assigned by a dedicated function. This function returns a time frame by which the following position request to the GPS sensor is delayed (line 4–6). The design and implementation of this function is subject of the next section.

Algorithm 9 Modifications to the sensing engine for adaptive positioning.

```
1: procedure SensingEngine
2:     p ← getPosition()
3:     while TRUE do
4:         posInterval ← getAdaptivePosInterval(Q,p)        ▷ Modified
5:         sleep(posInterval)
6:         p ← getPosition()
7:         ...                                              ▷ As in basic system
```

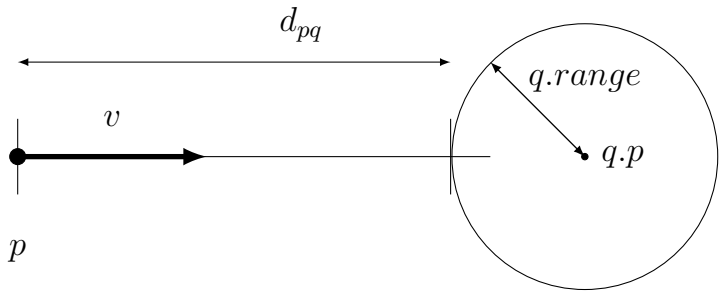


Figure 6.2: Basic idea of the adaptive positioning approach.

6.2.2 Adaptive Positioning Interval

The goal of the getAdaptivePosInterval function (Algorithm 9, line 4) is to return the maximal time frame that still ensures that no sensing range is passed while the sensing engine is in sleep mode (line 5). The idea for finding this value is as follows (see Figure 6.2): Given are the current position p of a device and a sensing query q. The distance between p and the sensing range of q is denoted as d_{pq} . If we assume that the device moves directly towards q with a constant velocity of v, the time frame t that the device needs to travel distance d_{pq} is given by the following formula:

$$v = \frac{d_{pq}}{t} \Leftrightarrow t = \frac{d_{pq}}{v} \tag{6.1}$$

With the help of this formula, a device can calculate the minimum time that it needs to reach the closest sensing range and delay the next position request accordingly.

Algorithm 10 shows the implementation of this idea. In the first step (line 2), the distance to the closest sensing range is calculated (referred to as d_{min}). Next, the device calculates an estimate about its future velocity v (line 3). Both values are calculated by separate functions, which will be explained in more detail later on. In line 4, the values d_{min} and v are used to calculate the minimum time t that is needed to reach the nearest sensing range. If the energy characteristics of the GPS sensor would not be considered, the value of t could be returned as

Algorithm 10 Calculation of the adaptive positioning interval.

```
1:  procedure getAdaptivePosInterval(Q, p)
2:      d_min ← getDistanceToQuery(Q, p)
3:      v ← getVelocity()
4:      t ← d_min / v
5:      if t ≤ delay_off then
6:          return t
7:      else
8:          t_r ← t − delay_req
9:          if t_r ≤ delay_off then
10:             return delay_off
11:         else
12:             return t_r
13:         end if
14:     end if
15: end procedure
```

result of the algorithm. However, as we have seen in Section 6.1, the GPS sensor has a non-linear energy characteristic that needs to be taken into account when calculating the adaptive positioning interval. More precisely, if t is smaller than the power-off delay of the GPS sensor, then t is returned (line 6). In this case, no energy can be saved since the sensor does not have enough time to power down. Note that even if the sensor cannot power down, it is more energy efficient to return a large value for t, since this delays the execution of the sensing engine and, thus, reduces its computational overhead. If t is bigger than the power-off delay, the sensor can power down before reaching q, but then needs the length of the request delay to provide a new position fix. Hence, t is reduced to t_r by subtracting $delay_{req}$ (line 8). If this reduction results in t_r being smaller than the power-off delay, $delay_{off}$ can be returned as result (line 10). In this case the sensor does not power down, which means there is no request delay for the next position fix. In contrast, if t_r is bigger than $delay_{off}$, the value of t_r is returned as result (line 12).

To calculate the distance to the closest sensing range d_{min} and the velocity v,

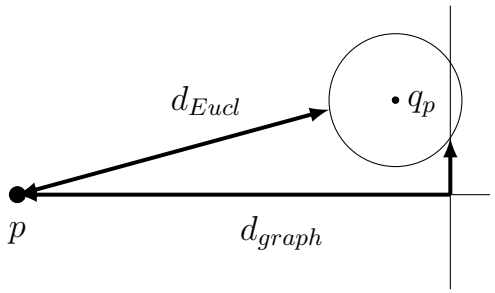


Figure 6.3: Euclidean distance vs. graph-based distance when measuring the distance between device and sensing range.

Algorithm 10 calls two subroutines (lines 2–3). Next, we have a look on how these subroutines are realized.

Distance to Sensing Query. The distance from the mobile device to the closest sensing range can be measured according to two different distance metrics, which are the Euclidean distance or the graph-based distance (see Figure 6.3). While the Euclidean distance measures the direct distance to the sensing range, the graph-based distance depicts the shortest distance to the sensing range on the road graph. In general, the graph-based distance is at least as long as the Euclidean distance and in most cases much longer. According to Equation 6.1, using the graph-based distance allows for longer sleeping times between two position requests and, hence, results in higher energy efficiency. However, the graph-based distance also requires a device to run a shortest path algorithm for measuring distances. Since the computational overhead for running such an algorithm is higher than calculating the Euclidean distance, there is a trade-off between energy for computation and energy savings for positioning. To analyze this trade-off, we next look at the distance calculation in more detail.

Calculating the distance to the closest sensing range according to the Euclidean distance is straightforward: The device compares is current position p against the positions of all known sensing queries and returns the closest distance. Algorithm 11 shows the implementation of this function. Note that the range of the respective sensing query must be subtracted (line 4), since the device should already have a position available when it enters the sensing range (see

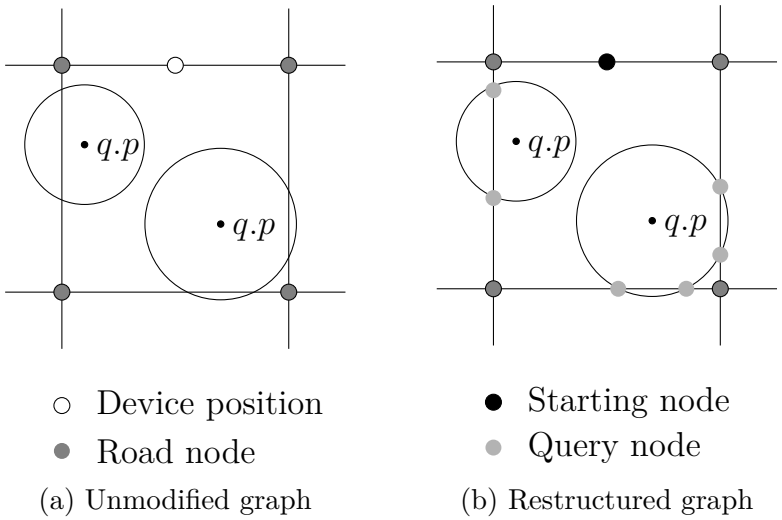
Figure 6.2). Since the calculation of the Euclidean distance requires constant time, the computational overhead for this algorithm is linear with the number of known queries, i.e. the time complexity is $\mathcal{O}(|Q|)$.

Algorithm 11 Calculation of Euclidean distance to closest sensing query.

```
1: procedure getDistanceToQuery(Q, p)
2:     d_min ← ∞
3:     for all q ∈ Q do
4:         d ← euclideanDist(p, q.p) − q.range
5:         d_min ← min(d, d_min)
6:     end for
7:     return max(d_min, 0)
8: end procedure
```

In contrast, the calculation of the graph-based distance is more complex. One prerequisite for this calculation is that a device knows the road graph on which it is moving (otherwise the device has to use the Euclidean distance metric). Before the distance calculation starts, the device position and all sensing ranges need to be mapped to the road graph. In general, a sensing range can intersect with more than one edge of the road graph (see Figure 6.4a). Each intersection point marks a starting point for reading data for the respective sensing query. Hence, the device has to determine the distance to the intersection point that is closest to its position. To avoid computing overhead on the devices, these intersection points can be determined by the server in advance and can be sent to the devices together with the sensing query. For running a shortest path algorithm, the road graph on which the device is moving is manipulated as follows (see Figure 6.4b): The current device position is inserted as starting node for the shortest path algorithm. For every intersection point of the graph with a sensing range, a query node is inserted.

A naive way of finding the distance to the closest sensing range is to replace line 4 of Algorithm 11 with the distance that is returned when executing a shortest path algorithm to find the distance between p and a query q. However, this would require to run the shortest path algorithm $|Q|$ times. For instance,



(a) Unmodified graph (b) Restructured graph

Figure 6.4: Restructuring the road graph for calculating the graph-based distance with a shortest path algorithm.

using Dijkstra's algorithm as shortest path algorithm would result in an overall time complexity of $\mathcal{O}(|Q||E| + |Q||V|log|V|)$, where $|V|$ is the number of nodes and $|E|$ the number of edges in the road graph. Due to the computational overhead of such an algorithm, the device may spend more energy on calculation than it can save through positioning. Hence, the following introduces a way to reduce the calculation overhead for the graph-based distance calculation.

The basic idea is to start the shortest path algorithm from the starting node and to stop its execution as soon as the first query node is found. Instead of running the shortest path algorithm $|Q|$ times, this only requires to run it once. Algorithm 12 shows the implementation of this idea. This algorithm is based on the Dijkstra implementation presented in [FT87] which utilizes priority queues to yield a time complexity of $\mathcal{O}(|E|+|V|log|V|)$. As discussed earlier, the device position p and the known queries Q are mapped to the road graph (line 2). This returns the starting node v_s and the modified road graph with vertex set V^*. The rest of the algorithm is identical to the standard Dijkstra implementation, with one exception: When a vertex is taken from the remaining nodes set N, the algorithm checks whether this vertex is a query node (lines 12–14). In this

Algorithm 12 Calculation of graph-based distance to closest sensing query.

```
1: procedure getDistanceToQuery(Q, p)
2:     V*, v_s ← mapToRoadGraph(Q, p)
3:     d[v_s] ← 0
4:     N ← {v_s}
5:     for all v ∈ V* do
6:         d[v] ← ∞
7:         N ← N ∪ {v}
8:     end for
9:     while |N| > 0 do
10:        v_min ← arg min_{v∈N} d[v]
11:        N ← N\{v_min}
12:        if isQueryNode(v_min) then
13:            return d[v_min]
14:        end if
15:        for all neighbors u of v_min do
16:            if d[v_min] + euclideanDist(u, v_min) < d[u] then
17:                d[u] = d[v_min] + euclideanDist(u, v_min)
18:            end if
19:        end for
20:    end while
21: end procedure
```

case, the algorithm stops and returns the distance to this vertex. Following the semantics of the Dijkstra algorithm, it is guaranteed that the first vertex that is found in line 12 is the closest query node for the device.

Velocity of Device. Besides the distance to the closest sensing range, Algorithm 10 takes the velocity of the device as input to compute the adaptive positioning interval. In general, there are different possibilities for setting this value. To ensure that a device will not miss any sensing range, v can be set to the device's maximum speed v_{max}. As detailed in Section 5.3.1, we assume that each device is aware of its own maximum speed. Even if the device moves directly towards the closest sensing query, the query will then not be missed. However, if the device is actually moving slower than v_{max}, the GPS sensor is activated earlier than actually required. Hence, a more optimistic approach is to use a device's average speed v_{avg} instead of v_{max}. This decreases the number of position fixes but may also reduce the effectiveness of the sensing system. More precisely, a sensing range can be missed if the device actually moves faster than v_{avg} and reaches the sensing range before the GPS sensor is activated again.

To see the effects on the energy efficiency and sensing effectiveness of choosing v_{avg} over v_{max}, both values will be used in the evaluation.

6.3 Modified Query Listener

After having introduced the concept of adaptive positioning in the last section, we next look on how this concept affects other parts of the PS systems. This section shows how the query listener on the device needs to be adapted, while the next section introduces a modification to the position update protocol.

As introduced in Section 2.2, the query listener runs on the mobile device and is part of the basic sensing system. It is responsible for receiving query messages from the server, which contain sensing queries that should either be added or removed from the set of known queries on a device. In order to use adaptive positioning, the query listener algorithm needs to be modified. To illustrate the necessity of this modification, we assume the following scenario: At time t, a

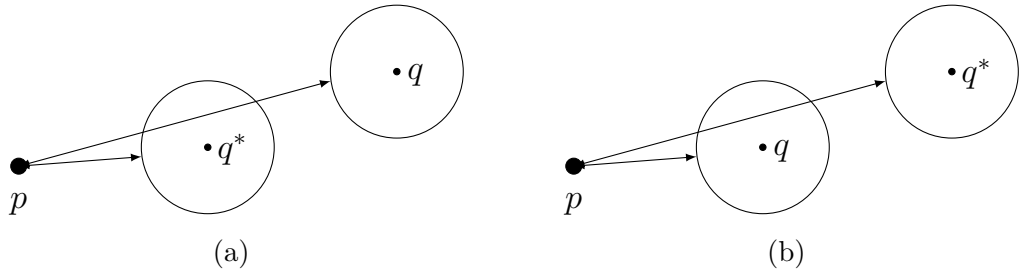


Figure 6.5: Arrival of a new sensing query.

device receives its current position p from the GPS sensor. Since the range of the closest sensing query q is some distance away, it decides to delay the next position fix for 100 s and puts the sensing engine to sleep mode (Algorithm 9, line 5). At time $t + 10\,\text{s}$, the device receives a new query q^* from the server. Now the following two cases can be differentiated:

1. Query q^* is closer to p than q (see Figure 6.5a). In this case, the sleeping time of the sensing engine needs to be reduced, otherwise the new query might be missed. The details of this reduction are given later on.

2. Query q^* is further away from p than q (see Figure 6.5b). In this case, the sleeping time of the sensing engine needs no adaption, since q is still the closest query to p.

Apart from these cases, a device can also receive a query revoke message while the sensing engine is in sleep mode. In this case, the sleeping time of the sensing engine can be extended if exactly that query is revoked that was previously the closest query when calculating the sleeping interval.

To consider all these cases, Algorithm 13 shows the modifications that need to be done for the query listener. While the basic parts of the query listener stay unmodified (lines 1–8), lines 9–11 include the additional concepts. In line 9, an adaptive positioning interval t is calculated. The input for this calculation is the query set Q that was modified in lines 4–8 and the last available device position p. In line 10, the result of this calculation is used to determine a new sleeping

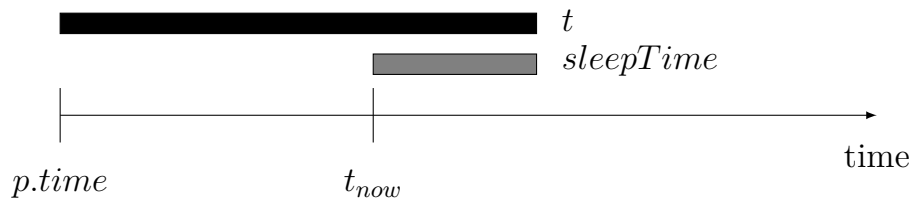


Figure 6.6: Time frames for calculating the sensing engine's sleeping time.

time for the sensing engine. To this end, the time frame between the current time t_{now} and the time of the last position $p.time$ is subtracted from t, since this is the time that the sensing engine already spent in sleep mode (see Figure 6.6). Note that this value can be negative if the newly added sensing query is close to p. Hence, a max function is used to prevent a negative sleeping time. With the resulting value, the sleeping time of the sensing engine is adjusted (line 11). In case $sleepTime = 0$, the sensing engine immediately wakes up and requests a position fix.

Algorithm 13 Modified query listener to adjust sleep time of sensing engine.

```
 1: procedure QueryListener
 2:     while TRUE do
 3:         msg ← receive()
 4:         if msg instanceOf(queryMsg) then
 5:             Q ← Q ∪ {msg.q}
 6:         else if msg instanceOf(stopQueryMsg) then
 7:             Q ← Q\{msg.q}
 8:         end if
 9:         t ← getAdaptivePosInterval(Q, p)          ▷ Begin of extension
10:         sleepTime ← max(t − (t_now − p.time), 0)
11:         adjustSleepTime(sleepTime)                ▷ End of extension
12:     end while
13: end procedure
```

6.4 Modified Update Protocol

With adaptive positioning a device does not periodically track its position since the GPS sensor is not always active. However, the availability of an up-to-date device position is a requirement for the position update protocol that was introduced in Chapter 4. More precisely, the update protocol needs the current position to constantly check if a position update needs to be sent to the PS server. In order to also work with adaptive positioning, this section introduces a modification to the previously presented update protocol.

In Chapter 4, two algorithms were introduced which are responsible for sending position updates to the server, namely the *basic update protocol* (see Algorithm 4) and its *opportunistic extension* (see Algorithm 5). The basic protocol waits until there is a new GPS position available and then checks whether this position violates the update condition. If this is the case, a position update is sent. In fact, this protocol also works when using adaptive positioning with the only difference that the position accuracy of the update protocol can no longer be guaranteed. For instance, the update condition can be violated while the GPS sensor is currently in idle mode. This violation can only be detected once the GPS sensor is active again. However, for the implementation of efficient position sensing, this protocol is not modified and the resulting inaccuracy is tolerated.

While the basic update protocol stays unmodified, the opportunistic extension of the update protocol needs minor modifications. The problem is that it queries the GPS sensor for the current device position whenever there is background traffic on the device. Since the sending of an opportunistic update does not bring any additional benefit when no new GPS position is available, we decide to not send an opportunistic update when the GPS sensor is currently deactivated.

Obviously, these modifications imply that the accuracy of the device position on the PS server can no longer be guaranteed by the update protocol. On the one hand, the resulting drop of the position accuracy has a negative effect on the efficiency of the query distribution, as we have seen in the evaluation of Chap-

ter 5. On the other hand, by temporarily disabling the GPS sensor a significant amount of energy can be saved on the device. We will see in the following evaluation how this trade-off between position accuracy and positioning energy affects the total energy consumption of the mobile device.

6.5 Evaluation

In this section, we look on the evaluation results of the efficient position sensing approach. First, the energy efficiency of adaptive positioning is evaluated and, subsequently, its effects on the sensing effectiveness of the PS system are analyzed. To see the system's relative performance, the aforementioned metrics are shown with respect to the basic sensing system in which devices continuously run the positioning system. Finally, the concepts from the efficient query distribution are included in this comparison in order to analyze the total energy savings when using all concepts that were presented so far.

6.5.1 Simulation Setup

The evaluation in this section is based on the same system setup that was used for the query distribution in Section 5.6. Hence, all concepts were implemented with the help of the ONE simulator and, if not stated otherwise, are based on the same standard parameter configuration that is used in Section 5.6. The energy that is needed for calculating distances to sensing ranges (see Section 6.2) is measured as follows: Parts of the simulation were executed on a Samsung Galaxy Nexus i9250 to log the time that is needed by the device for the distance calculation. Furthermore, the power consumption of the device during calculation was measured by attaching a measurement kit between the device's battery and the device. The results of these measurements show that a device consumes on average 1.3 W while running the distance calculation. This value was used in combination with the logged execution times to calculate the computation energy in the simulation. In the following, this energy is included in the positioning energy. To measure the energy for position fixes, the energy values and

state transition delays from Section 6.1 were used.

For the evaluation, we compare three different algorithms:

basic Implements the basic sensing system that uses continuous positioning with a positioning interval $c_p = 1\,\text{s}$ (as in [KLG+09]).

eucl Implements adaptive positioning using the Euclidean distance for measuring distances to sensing ranges.

graph Implements adaptive positioning using the graph-based distance for measuring distances to sensing ranges.

Since the distance calculation can either take the average or the maximum speed as base for calculating the GPS timeout, the corresponding speed is attached to the name of the approach. For instance, *eucl-max* denotes the adaptive positioning using the Euclidean distance and the maximum speed for calculating the GPS timeout.

6.5.2 Energy Efficiency of Adaptive Positioning

First, we have a look at the energy that is consumed by the mobile devices for positioning. Figure 6.7 shows on the y-axis the positioning energy that is consumed over all mobile devices. We can see that the basic approach consumes significantly more energy than the optimized approaches in all scenarios. For instance, the graph-based approach that uses average speed saves 90 % on average compared to the basic approach.

To see the differences between the optimized approaches in more detail, Figure 6.8 leaves out the basic approach and only shows the optimized ones. The figure shows that the use of the maximum speed consumes more energy than using average speed, no matter which distance metric is used. For the pedestrian traces, the maximum walking speed is 40 % higher than the average speed of a pedestrian. Using maximum speed, the graph-based approach is more energy efficient than using the Euclidean distance. More precisely, the graph-based approach saves 5 % of energy compared to the Euclidean distance approach.

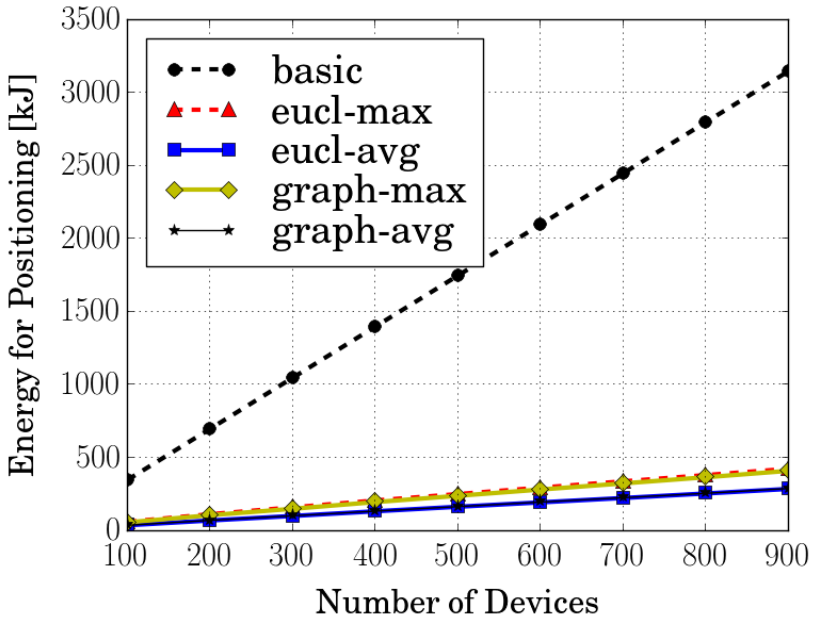


Figure 6.7: Positioning energy for different number of devices.

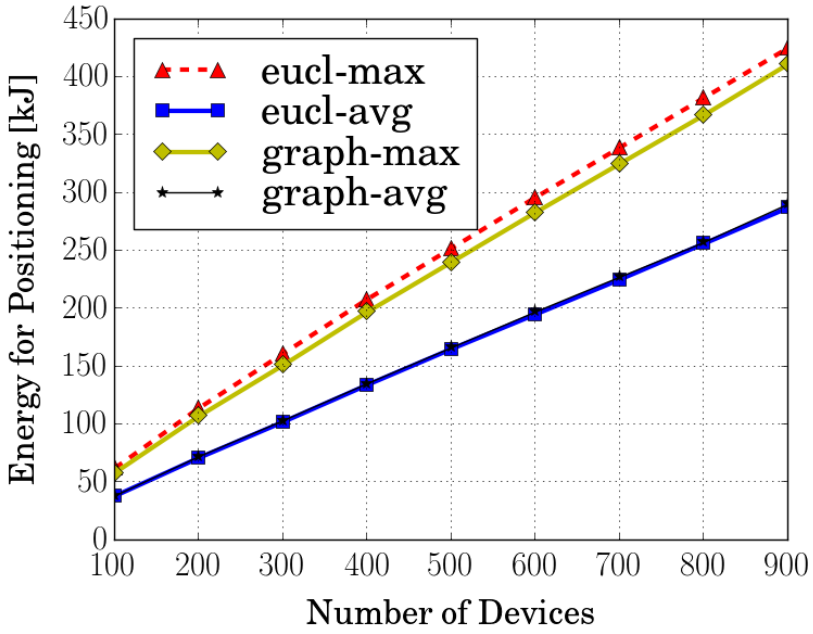


Figure 6.8: Positioning energy for different number of devices.

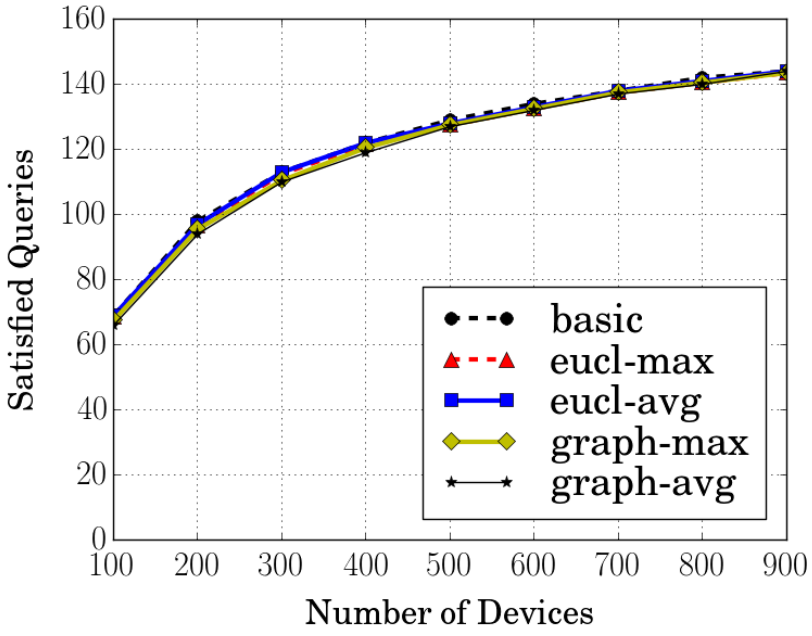


Figure 6.9: Number of satisfied sensing queries for different number of devices.

However, when using average speed there is no difference between both approaches, i.e., both consume on average the same amount of energy. This can be explained as follows: When using a device's average speed for calculation, the resulting GPS timeout is longer as when using maximum speed. Hence, it is more likely that a device receives a new sensing query in this time frame that wakes up the GPS (see Section 6.3). In this case, there is no difference between the Euclidean and the graph-based approach considering the length of the GPS timeout. Furthermore, also considering the computational overhead for calculating the graph-based distance, the graph-based approach consumes the same amount of energy as the distance-based approach.

6.5.3 Effectiveness

The energy results show that the use of adaptive positioning gives rise to tremendous energy savings with respect to positioning energy. Next, we will check if these savings come for the price of decreased sensing effectiveness. For this purpose, the number of satisfied sensing queries is shown in Figure 6.9. We can see

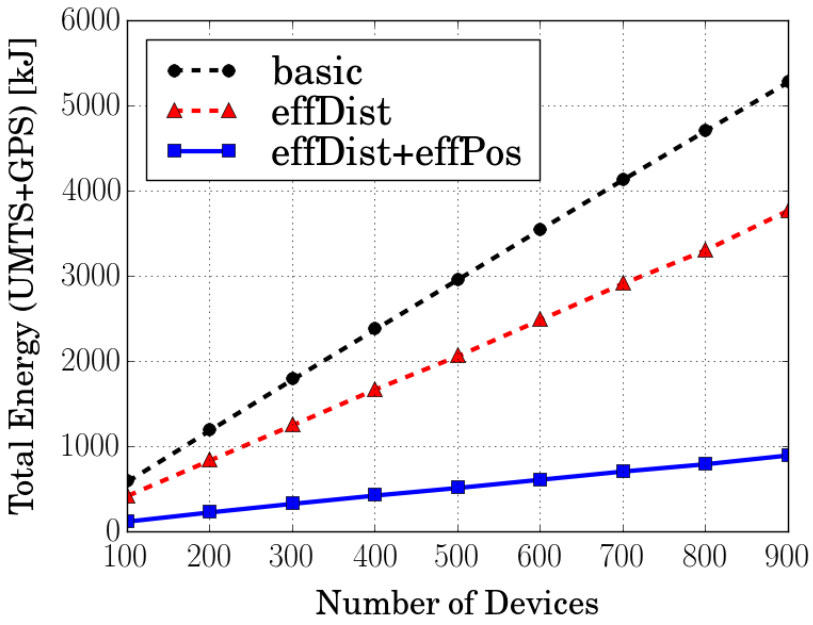


Figure 6.10: Total energy for different number of devices.

from the figure that the approaches lowers the sensing effectiveness only insignificantly (on average $< 1\,\%$). This means that even the use of average speed for the GPS timeout calculation is sufficient to not lower the sensing effectiveness. Hence, the use of this more optimistic speed estimation pays off, as it does not decrease sensing effectiveness but increases energy efficiency, as we have seen before.

6.5.4 Total energy efficiency

Finally, we have look at the combination of the different approaches that were introduced in this thesis so far. For this purpose, we look at a device's total energy consumption that includes communication energy and positioning energy. Figure 6.10 shows the result of the basic approach, an approach that only uses efficient query distribution (*effDist*) and an approach that uses efficient query distribution together with efficient position sensing (*effDist+effPos*). Here, the latter one uses the graph-based distance metric and the average speed for calculating the GPS timeout. While the efficient query distribution already increases

the energy efficiency of the basic approach, we see that the additional use of adaptive positioning results in big additional energy savings. Compared to the basic approach, the energy savings of this combined approach are on average 82 %.

6.6 Related Work

Before concluding this chapter, we have a look on related work in efficient position sensing. Taking the context of the user into account, Ryder et al. [RLR+09] and Lu et al. [LYL+10] propose a way to increase the battery lifetime of a mobile device by adapting the GPS sampling rate to the movement mode of the user. For instance, no position fix is needed while the accelerometer indicates that the user stays at the same position. However, Chon et al. [CTS+14] have recently shown in a measurement study that this strategy does not reduce the energy consumption of a mobile device. Instead, continuous sampling the accelerometer prevents a device's CPU from going into sleep mode. Hence, the proposed strategy actually increases the total energy consumption.

Other systems follow the same approach that is used in this chapter and adaptively schedule the GPS sensor. For instance, the RAPS system [PKG10] adaptively turns on the GPS sensor if the position uncertainty of a device exceeds a certain accuracy threshold. Similar approaches have been proposed to efficiently track the position of a device. Farrell et al. [FLR07; FRC11] show in their work on continuous range queries that the next position fix can be deferred until the earliest point in time when the mobile device can reach the boundary of the range query. Based on this work, Kjærgaard et al. propose the EnTracked system [KLG+09] that implements the aforementioned concepts on a real device and give valuable insights to the power characteristics of a mobile device positioning. While these approaches temporarily disable GPS positioning until a certain inaccuracy threshold is reached, the approach presented in this chapter follows a different idea. More precisely, the presented approach does not limit the inaccuracy of the device position but ensures that an accurate position is

available before reaching a sensing range. As a result, a different strategy for scheduling the GPS sensor is required.

6.7 Conclusion

The evaluation in this chapter showed that continuously running a positioning system consumes a large amount of energy on the mobile device. To account for this fact, the concept of adaptive positioning was introduced that temporarily disables the GPS sensors. However, disabling the positioning system can also result in missed sensing queries, since a device can only read data for a sensing query when GPS is active. This chapter introduced a concept that tackles both, increasing energy efficiency while not decreasing sensing effectiveness. By adaptively calculating an individual GPS timeout, depending on the current device position and the position of sensing ranges, the GPS can be disabled long enough to save energy, but not too long to miss any sensing query.

7 Efficient Indoor Position Sensing

The concepts that were introduced so far assume that mobile devices are moving outdoors while capturing sensor data for the PS system. In fact, many sensing scenarios like air pollution monitoring require sensor data that was captured outdoors. Besides these outdoor scenarios, indoor environments can also be subject of sensing scenarios that provide valuable insights. This is especially true if we consider that people typically spend large fractions of their time indoors [Ott89]. For instance, indoor air quality monitoring [JLT+11] and automatic floor plan generation [PBD+14] are only two examples out of a wide range of possible indoor application scenarios. To account for this fact, this chapter extends the concepts that were presented so far to be applicable in indoor environments. On the one hand, this does not require any changes to the opportunistic position update protocol and the efficient query distribution. Both can receive the required position information from the inertial positioning system (IPS) that runs on the device. On the other hand, the previously introduced concept of efficient position sensing cannot be applied without major modifications, as we will see in the following.

As already discussed in the system model, devices rely indoors on an IPS. Similar to GPS, an IPS has high power consumption and therefore drains a device's battery quickly. Thus, constantly running an IPS lowers the user acceptance of the PS system, which has a negative impact on the coverage of the sensing system. Similar to the concept of efficient outdoor position sensing, a promising way to save energy is to temporarily turn the IPS off when no sensing query is close to the device. However, this is not straightforward since an IPS requires an absolute starting position when it is turned on. More precisely, an IPS is based on angular and linear acceleration values, which are integrated to

calculate a motion vector. This vector is added to an absolute starting position, which is not available when the device is already indoors and decides to start the IPS. Hence, to enable efficient position sensing also for indoor scenarios, these special characteristics of an IPS need to be taken into account. To address these problems, this chapter introduces concepts that allow to temporarily turn off the positioning system in indoor scenarios. As a result, the energy efficiency of indoor positioning can be increased, however, to achieve this we have to accept a decrease in sensing effectiveness of the PS system, as we will see later on.

The remainder of this chapter is structured as follows: Before the actual concepts are introduced, Section 7.1 gives an overview of the different components of the positioning approach. Subsequently, Section 7.2 introduces an energy model for the IPS and WiFi scanning, which will be used to support the IPS. Sections 7.3 and 7.4 introduce the main concepts to implement the efficient indoor position sensing. In Section 7.5 we look at the evaluation results for the presented approach, before Section 7.6 gives an overview of related work on this topic. Finally, Section 7.7 concludes this chapter.

7.1 Overview

The general idea of efficient indoor position sensing is the same as for outdoor positioning, which was introduced in the last chapter: Mobile device disable their energy-intensive positioning system as long as there is no sensing range in their current vicinity. As mentioned before, this is not easily possible with an IPS, since it needs an absolute starting position when being turned on. For this purpose, a device additionally uses a low-energy WiFi-based positioning system. The reference data for this system is collected on-the-fly and, hence, it does not require any prior collected WiFi fingerprints. This WiFi-based positioning is used to provide so-called anchor points, which serve as absolute reference points when turning the IPS on again. The remainder of this section gives an overview over all the components that are necessary to implement this approach. In subsequent sections, each component will be elaborated in more detail.

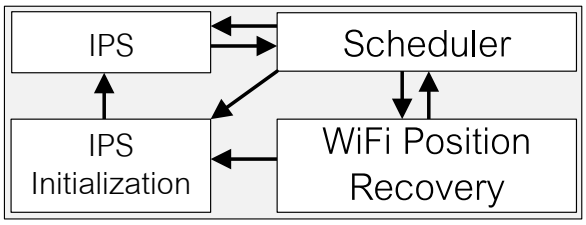


Figure 7.1: Components of the efficient indoor positioning approach.

In contrast to the previously presented concepts, the efficient indoor position sensing approach cannot be easily integrated into the basic sensing system by replacing a few lines of code. Instead, it consists of a set of different components that run on the mobile device in parallel to the basic sensing system. Figure 7.1 gives an overview of these components and their relations to each other. In general, two operation modes can be distinguished which are *IPS on* and *IPS off*. While the IPS is on, a mobile device runs the non-modified version of the sensing engine that was presented in Algorithm 3. In this case, the IPS provides the position that the sensing engine uses for checking if the device reached a sensing range. In contrast, if the IPS is off, the sensing engine on the device is temporarily disabled and the device cannot read sensor data.

The decision when to turn the IPS on or off is made by the *Scheduler* component, which takes the current device position and the locations of the sensing ranges as input. With the help of a mobility model, the scheduler predicts whether it is likely that the device will reach a sensing range in the near future. If that is the case, the IPS must be running when reaching this range. Hence, the scheduler uses a mobility model to decide when the IPS can be turned off and when the initialization of the IPS must be started again.

The *WiFi position recovery* is activated once the IPS is disabled. To ensure that the IPS can be initialized before reaching a sensing range, it is responsible for two tasks: First, it keeps track of the device position. Since the IPS is temporarily turned off to save energy, a device still needs to be able to locate itself. Otherwise it is possible that it passes the next sensing range without capturing sensor data. For this purpose the WiFi position recovery helps to

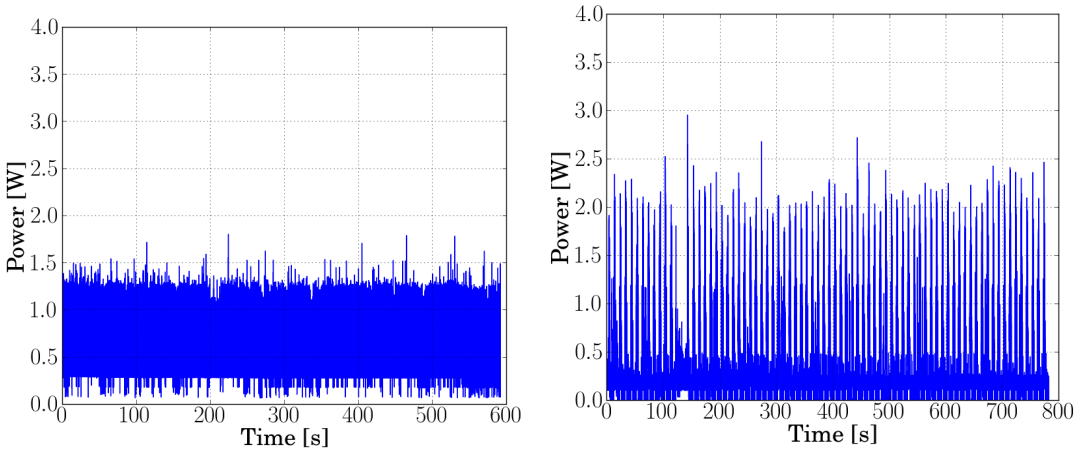
track the device position with decreased accuracy. Second, it helps to initialize the IPS after the scheduler decided that the IPS should be turned on. To support these operations, it uses so-called *WiFi anchor points*, which provide coarse device positioning with small energy overhead. These anchor points are placed inside a building and are associated with a known position and a set of WiFi fingerprints. Such a fingerprint consist of several *received signal strength indicator* (RSSI) measurements of WiFi signals that a device can receive at the anchor point's position. Devices can use these RSSI values as reference data for finding to the closest anchor point in order to get a coarse position, as we will see later on in more detail.

The *IPS initialization* component interacts with the WiFi position recovery to allow for dynamically turning on the IPS. To provide an absolute starting position for the IPS, it uses WiFi anchor points to pin a relative IPS trace to the absolute position of an anchor point. To precisely know when a device passes such an anchor point, acceleration data from the IPS is used to identify sharp turns at corners (hallway intersections and turns). Thus, anchor points will be strategically placed at each corner such that the trace can be precisely aligned when the device reaches a corner and makes a turn.

7.2 Energy Efficiency of IPS and WiFi Positioning

One prerequisite to increase energy efficiency with the aforementioned approach is that scanning for WiFi fingerprints consumes less energy than running the IPS. To backup this assumption, we look in this section at the results of an experiment that was conducted to compare the energy consumption of an IPS and WiFi positioning.

For this experiment, both positioning methods were implemented on a Samsung Galaxy Nexus i9250 and a measurement kit was attached between the battery and the device to measure the device's power drain. To implement the IPS system, an Android application was developed that runs an IPS according to the system of Li et al. [LZD+12]. This application reads sensor data from the



(a) IPS (b) WiFi

Figure 7.2: Power consumption of the mobile device over time: (a) Device running the IPS. (b) Device performs WiFi scans every 10 s.

device's compass, gyro and accelerometer and fuses this data to detect individual steps of the device user along with the device's current heading. For the WiFi positioning, an application was implemented that periodically performs a WiFi scan every t_{scan} seconds and then compares the scan results against a database of WiFi fingerprints.

The results of this measurement are shown in Figure 7.2. The power consumption for the IPS is shown in Figure 7.2a, while the results for the WiFi positioning with a scan period of $t_{scan} = 10\,\text{s}$ are shown in Figure 7.2b. By looking at the power consumption over time for the IPS, we see that running the IPS keeps the device constantly active. This can be attributed to the fact that the device has to continuously sample its sensors and analyze the obtained data. Therefore, the CPU cannot go into the less power consuming idle mode. For the WiFi positioning, we see that the peak power consumption when a WiFi scan is performed is higher than the power consumption of the IPS. However, between two WiFi scans the device consumes less power than the IPS. In this time frame, the device's CPU is able to power down to the energy-saving idle mode. To see which of these two systems is more energy efficient in total, Figure 7.3 shows the

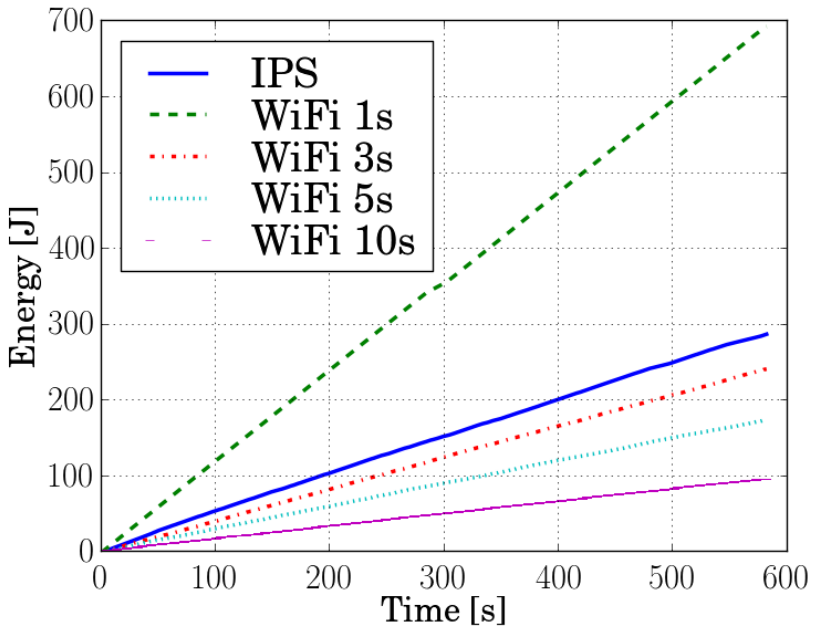


Figure 7.3: Energy consumption of IPS and WiFi positioning over time.

cumulated energy consumption for both approaches. In addition to the WiFi scan period of $t_{scan} = 10\,\text{s}$, this figure also shows the energy consumption of the WiFi application for a scan period of $t_{scan} = \{1, 3, 5\}\,\text{s}$. We see that the WiFi positioning system is more energy efficient than the IPS if the scan period is bigger than one second. However, if a WiFi scan is performed every second, the total energy consumption is higher than for the IPS. Hence, periodic WiFi scanning with an one second interval has an average power consumption of 1.19 W and, thus, is more energy consuming than running the IPS that has an average power consumption of 0.46 W. In contrast, using a larger scan period can save a lot of energy. For $t_{scan} = 3\,\text{s}$, the WiFi approach consumes 0.41 W, which is 89 % of the IPS power. For $t_{scan} = 5\,\text{s}$, the power consumption is 0.3 W (= 65 % of IPS) and for $t_{scan} = 10\,\text{s}$ the WiFi approach consumes only 0.16 W (= 35 % of IPS).

This study shows that it is worth turning off the IPS and using WiFi positioning instead. On the downside, using WiFi positioning with a sample period of 10 s seconds can provide only coarse-grained position accuracy. However, this

is sufficient if the device is not close to a sensing range, as we will see in the following sections. For the rest of this chapter, t_{scan} is set to 10 s. In the next section, we will see how WiFi positioning is set-up.

7.3 WiFi Position Recovery

The WiFi position recovery is based on the concept of geographically distributed anchors points and follows the idea to supply each device with information to identify these points when passing them. This information includes the geographic position of each anchor point and the RSSI values that were measured at these points during the anchor point setup phase. After the IPS has been turned off, a device can use this information to estimate its position by comparing the results of WiFi scans with the known RSSI values of the anchor points. More details on this comparison are given later on.

Before anchor points can be used for positioning, they first need to be set-up. Since we assume no prior knowledge about the distribution of WiFi signals inside the building, anchor point setup has to be performed "on-the-fly" in an initial phase.

For choosing suitable positions for the anchor points inside the building, three requirements must be fulfilled:

1. The total number of anchor points inside a building must be small. They should only be set up at few locations, since for every anchor point WiFi RSSI data needs to be collected by the devices, which is energy consuming (as shown in Section 7.2).

2. The locations of anchor points should be chosen in such a way that they allow devices to find their coarse position from few WiFi scans. More precisely, if the IPS is off, only a few WiFi scans should be sufficient for a device to identify the closest anchor point.

3. Finally, the locations of the anchor points should be selected in such a way that they can provide a precise position for initializing the IPS. More

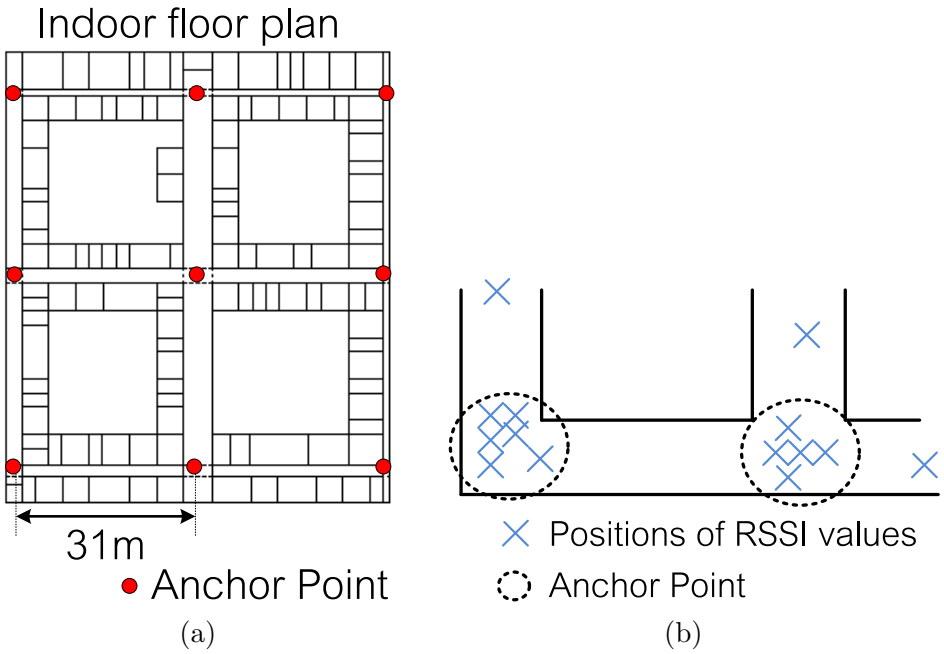


Figure 7.4: Building the WiFi anchor point database: (a) Locations of the WiFi anchor points. (b) Recording and filtering of RSSI values.

precisely, an anchor point should be placed in such a way that it is possible to indicate from an IPS trace at which position in the trace the anchor point was passed. Since this is a rather abstract requirement, we will see in the following how this can be implemented.

To meet these requirements, anchor points are only placed at hallway intersections as shown in Figure 7.4a. Since there are typically only a few hallway intersections in a building, the number of anchor points and necessary WiFi scans is limited. Moreover, the position of intersections can be identified easily by looking for sharp turns of about 90 ° in the IPS trace. Hence, for setting-up the anchor points, devices perform WiFi scans whenever they make sharp turns and upload the measured RSSI value together with the current device position to the PS server. The PS server uses the uploaded RSSI values and associates them with an anchor point according to the position where the RSSI was recorded. Note that RSSI values can actually be recorded aside from intersections if the

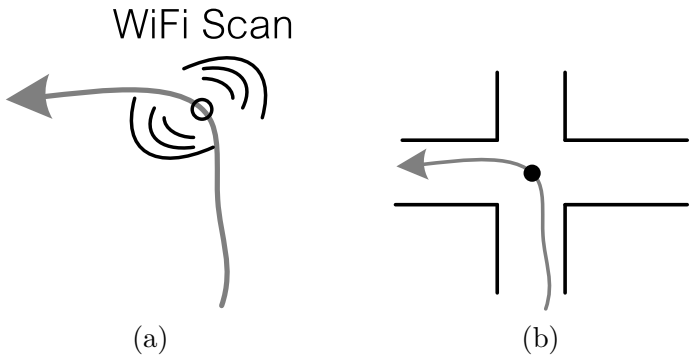


Figure 7.5: Reposition of a relative IPS trace: (a) Recording a WiFi scan at a 90° turn. (b) Mapping the relative trace to the center of the according intersection.

user made a sharp turn at some other place, for instance, when entering a room. These values can be filtered out (see Figure 7.4b) by the PS server by using map knowledge. The result is a list of anchor points, each one of them associated with a set of attached WiFi RSSI values. This list is communicated back to the devices when the server distributes sensing queries to the devices.

After setting-up the anchor points, they can be used by devices for coarse positioning while the IPS is turned off. To this end, a device performs a WiFi scan every 10 s and identifies its closest anchor point. This is done by selecting the anchor point whose RSSI fingerprint is closest to the RSSI values measured by the device using the Euclidean distance as metric. It has been shown that this approach has an average accuracy of about 4 m [LGY+12]. In general, hallway intersections in indoor environments are further apart than 4 m. A smaller distance between intersections would imply that there are rooms in between that are even smaller than this distance, which is not a realistic assumption. As a result, a device is able to identify the correct anchor point with high accuracy. We will see in Section 7.5 the results of an experiment that confirms this accuracy. With the help of the anchor points, a device is able to identify its closest intersection point and to track its current path through the building. This information serves as input for the position scheduler that is presented in Section 7.4.

In general the layout of the building influences the number and the distribution of the anchor points. While the building layout in Figure 7.4a has only a very limited number of intersections that are far apart from each other, there are other building layouts with much more intersections. The presence of more intersections results in more anchor points, which allows a device to track its position more accurately with WiFi. However, WiFi positioning can result in errors if there are many intersection that are close too each other. In this case, the closest intersection may not be clearly identified and a position shift may occur. We will see in the evaluations later on that the distance between anchor points should be bigger than 5 m. Considering the average room sizes of buildings, this precondition holds true for almost all room layouts. On the other hand, in layouts in which intersections are sparse, anchor points should be set at every intersection as shown in Figure 7.4a.

Besides position tracking, anchor points can also be used as absolute starting positions when initializing the IPS. More precisely, a device can turn on its IPS to record a relative trace without having a starting position for this trace. It then has to wait for a 90 ° turn as indicated by the gyroscope (angular acceleration) values, and perform a WiFi scan at this point. By comparing the obtained RSSI values with the fingerprints of the anchor points and selecting the point with the best matching fingerprint, the device knows its precise location, which is the location of the identified anchor point (see Figure 7.5a). With the help of this information, the device can position the relative IPS trace recorded so far and map it to the absolute center of the intersection (see Figure 7.5b). As a result, the IPS trace has an absolute starting point and can be used to continue indoor positioning.

Note that there can be more than one possibility to match the IPS trace to the center of the intersection. For instance, the trace in Figure 7.5b would also fit if it starts from the top hallway and moves to the right hallway. To overcome this issue, the device can use information about its previous movement path to find the correct matching. This information can be provided by the anchor point tracking, as described earlier. After matching the IPS trace to the intersection,

the IPS initialization is finished.

One problem with the aforementioned approach is that the position of the trace might not be perfectly accurate if the device has not turned exactly on the center of the intersection. We investigate this effect further in the evaluation in Section 7.5. Furthermore, it is possible that the IPS indicates a 90 ° turn even though the device did not pass an intersection. This can be the case if the device moves into a room, for instance. Hence, to clearly identify that a device turns at an intersection, we also require that the trace before and after the 90 ° turn is straight for a certain minimum distance. To be sure that a device is not moving in a room after this turn, this distance must be longer than the depth of the longest room that is available in the building. For the evaluation of this approach, a distance of 5 m was used, which turned out to be sufficient to separate turns at intersections from other turns.

7.4 Position Scheduler

This section introduces the position scheduler algorithm, which decides when to turn the IPS off and when to start the initialization of the IPS again. In general, a device can turn its IPS off when no sensing range is nearby, while the IPS initialization must be finished before the device reaches a sensing range. To support this decision, the scheduler uses a mobility model for predicting the future path of the mobile device.

Next, we look at the assumptions that we make about this mobility model, before the decision making for the IPS is presented in detail.

7.4.1 Mobility Model

The design of realistic indoor mobility models is still an open research question [CBH+04; KKR11], which is beyond the scope of this thesis. Therefore, no specific mobility model is defined at this point and instead the mobility model is assumed to be a black box. Later in the evaluations, a specific model will be used to test the system. The only thing that we require from this mobility

model is that it can determine the probability $P(a \mid t)$ that a device will visit a certain area a, after having traveled along trace t. Based on $P(a \mid t)$, the scheduler then decides whether to stop or initialize the IPS.

7.4.2 Turning off IPS

To turn the IPS off, two conditions must be fulfilled:

1. The device is currently not moving directly towards any sensing range, i.e., the device has to pass at least one intersection before it can reach any sensing range.

2. The IPS initialization can be completed before reaching a sensing range.

A mobile device periodically monitors these conditions and only turns off its IPS if both conditions are fulfilled. The first condition can be easily checked by comparing the current IPS position of the device to the set of known sensing queries. In contrast, the second condition is harder to evaluate, as we will see in the following.

To initialize the IPS, a device must turn at an intersection by 90 ° before entering a sensing range. Otherwise the IPS initialization cannot be completed because there is no anchor point to which the relative trace of the IPS can be mapped to. To ensure that IPS initialization can be completed before reaching a sensing range, the device performs the following steps: When the IPS is on, a device determines towards which direction it is currently moving. We denote this intersection as I_n. Note that I_n is not fixed to a certain intersection. Hence, if the device owner turns and walks towards another intersection I_n is updated accordingly. In a next step, the set Q_{I_n} is defined that contains all sensing queries whose sensing ranges are reachable from I_n without a turn (see Figure 7.6). For any query $q \in Q_{I_n}$, intersection I_n is the last turning point if the device walks towards q via intersection I_n. Hence, the IPS is turned off if it is not very likely that the device moves to one of the sensing ranges in Q_{I_n}. In contrast, if the device is likely to move to one of these sensing ranges then the IPS should not

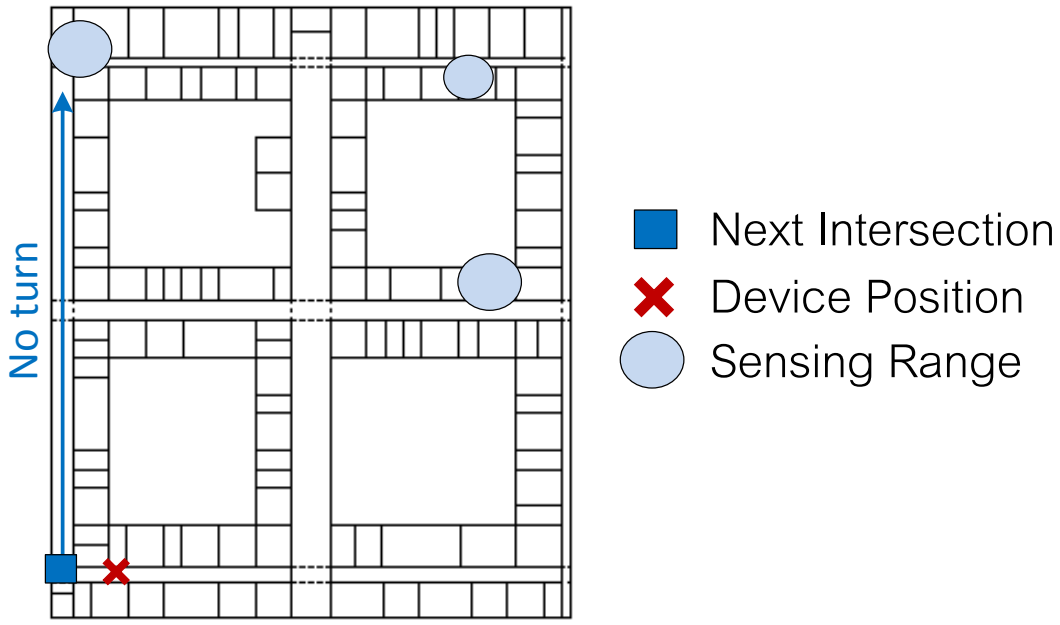


Figure 7.6: Decision on turning off IPS: Only the sensing range in the upper left corner can be reached without a turn from the next intersection that the device passes.

be turned off, since I_n would then be the last possibility for initializing the IPS again.

To estimate the probability that the device moves to one of the sensing ranges in Q_{I_n} the mobility model is used. More precisely, the mobility model provides $P(a_q \mid t)$ for every $q \in Q_{I_n}$, where a_q is the area that is covered by q. The probability that the device will move to at least one of these areas is then given as:

$$p_{LQ} = 1 - \prod_{\forall q \in Q_{I_n}} 1 - P(a_q \mid t) \tag{7.1}$$

Note that this function is a special case of the binomial distribution.

The result of Equation 7.1 can now be used to decide whether to turn the IPS off. For this decision, two different algorithms are presented in the following, which are compared to each other in the evaluation.

For the first algorithm, a system parameter c_p is used to decide whether the IPS stays on or will be turned off:

$$IPS = \begin{cases} off, & \text{if } c_p > p_{LQ} \\ on & \text{else} \end{cases} \tag{7.2}$$

The use of this algorithm has the advantage that the likelihood of turning the IPS off can be controlled by parameter c_p and, hence, the performance and energy efficiency can be adapted.

For the second algorithm, a random sample s_r from the uniform distribution between $[0, 1]$ is drawn and the decision whether the IPS stays on or will be turned off, is made as follows:

$$IPS = \begin{cases} off, & \text{if } s_r > p_{LQ} \\ on & \text{else} \end{cases} \tag{7.3}$$

The use of this algorithm has the advantage that no system parameter needs to be set. Instead the probabilistic characteristic of p_{LQ} is utilized to support this decision.

Once the IPS was turned off, the following algorithm determines when the IPS should be turned on again.

7.4.3 Turning on IPS

After the IPS has been turned off, it must be initialized again, before the device enters a sensing range. This is the same requirement that is described by Condition 2 in the last section. Hence, the same algorithm is applied to decide whether the IPS should be turned on and initialized. For this purpose, the device first determines towards which intersection I_n it is currently moving. Since the IPS is turned off, intersection I_n can be obtained by using the available information about the previously visited hallways. More precisely, knowing which hallways the device has recently passed and knowing its closest intersection from a recent

WiFi scan, a device is able to infer the intersection I_n to which it is currently moving to. With the help of this information, the device starts the IPS initialization if it is very likely that it moves to a sensing range that is reachable from I_n without a turn, as described in the last section. Note that also in this case, the closest intersection I_n is always updated and the decision whether to turn the IPS on is reevaluated accordingly.

7.5 Evaluation

The evaluation of the presented approach consists of two parts. The first part analyzes the accuracy of the WiFi position recovery and the IPS initialization. The second part evaluates the efficient indoor position sensing algorithm when it is used on top of the MapGENIE mapping system [PBD+14] that was developed in the ComNSense project. This enables us to see the energy efficiency of the approach and its impact on the sensing effectiveness. Before these two parts are described in detail, the experimental setup is introduced.

7.5.1 Experimental Setup

The evaluation of the system is based on the MapGENIE system and the according dataset [PBD+14]. MapGENIE uses a set of opportunistically collected indoor pedestrian traces to automatically derive an indoor floor plan. A more detailed explanation of MapGENIE is given in Section 1.4. The MapGENIE dataset is publicly available[1] and includes 154 odometry traces, which were collected from four volunteers with an Android smartphone in combination with a foot-mounted inertial measurement unit in order to implement an IPS. The total length of these traces is more than 22 km. For the following evaluation, the MapGENIE system was modified to execute the presented efficient position sensing algorithm while constructing the indoor floor plan. For this purpose, all areas of the indoor floor plan which are not accurately mapped according to a given quality metric (see [BPD+14]) are marked as sensing ranges. Moreover, a

[1] see `http://www.comnsense.de/downloads`

simple mobility model is used as input to the position scheduler that is described in [BPD+14]. This model assumes an uniform visiting probability for each area of the building.

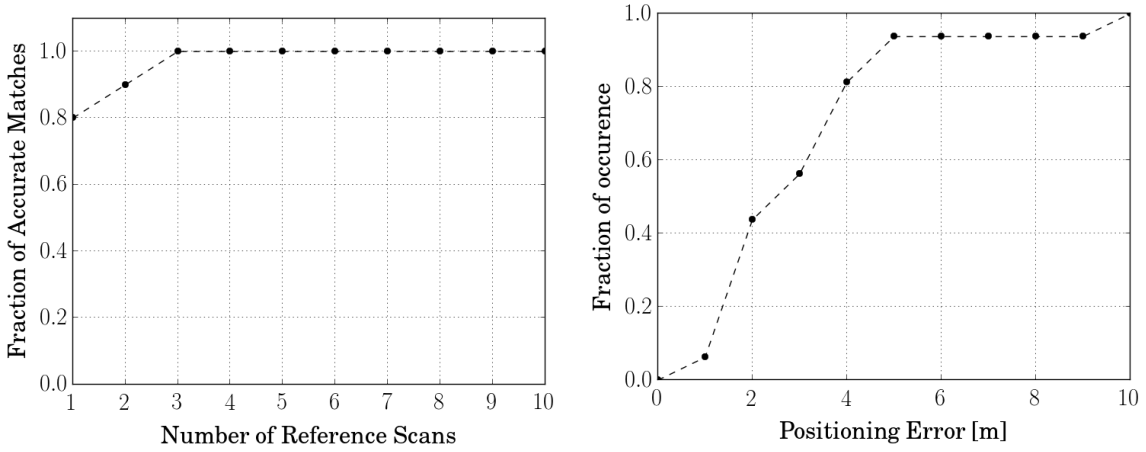
7.5.2 Positioning Accuracy

To evaluate the WiFi position recovery and the IPS initialization, in a first step WiFi RSSI data was collected to set-up the reference database. More precisely, WiFi scans were performed at intersection points in the same building in which the MapGENIE dataset was recorded (see Figure 7.4a). From this data, the WiFi anchor point database was set-up.

To evaluate the accuracy of the WiFi position recovery, WiFi scans at 30 arbitrary positions were performed in the building with an Android phone. For each scan, an anchor point from the database was assigned by comparing the scan's RSSI values and choosing the anchor point with the minimal Euclidean distance. The assignment is accurate if the intersection that is closest to the ground truth position is equal to the intersection that is associated with the assigned anchor point. Furthermore, we are interested in the number of WiFi scans that need to be present in the reference database in order to reach accurate anchor point identification. Hence, the accuracy of identifying the correct anchor point was evaluated when different numbers of scans per anchor point are stored in the database.

Figure 7.7a shows the success rate of this experiment. The y-axis shows the fraction of correctly identified anchor points while the x-axis shows the number of WiFi scans from the database that were used for identifying the closest anchor point. We see that if at least three WiFi scans are available in the database at an anchor point, the WiFi scan can be assigned to the right anchor point. Hence, only a small set of reference data is needed to implement accurate anchor-based WiFi positioning. The energy overhead for building the anchor point database is therefore negligible.

As mentioned before, IPS initialization cannot perfectly reestablish a position since it maps the relative IPS trace to the center of an intersection. If a device



(a) (b)

Figure 7.7: Position accuracy: (a) Accuracy of anchor point identification. (b) Cumulative distribution function of shift errors when resuming IPS positioning.

does not turn directly at the center of the intersection, the reestablished trace is shifted. Figure 7.7b shows a cumulative distribution function of this shift error in meters. From this plot we see that in 50% of the cases this error was smaller than 3 m and in 90% of all cases the error was smaller than 5 m. Hence, in most of the cases the IPS can be initialized without a large shift error.

7.5.3 Energy Efficiency

To evaluate energy efficiency and sensing effectiveness (in the next section) of the presented approach the following algorithms are considered in the evaluation:

basic This approach denotes the unoptimized execution of the MapGENIE mapping algorithm when all devices are running the IPS all the time.

effMap-thre This approach shows the energy for executing the mapping approach when using efficient position sensing with the threshold-based decision algorithm (see Equation 7.2).

effMap-rand This approach shows the energy for executing the mapping ap-

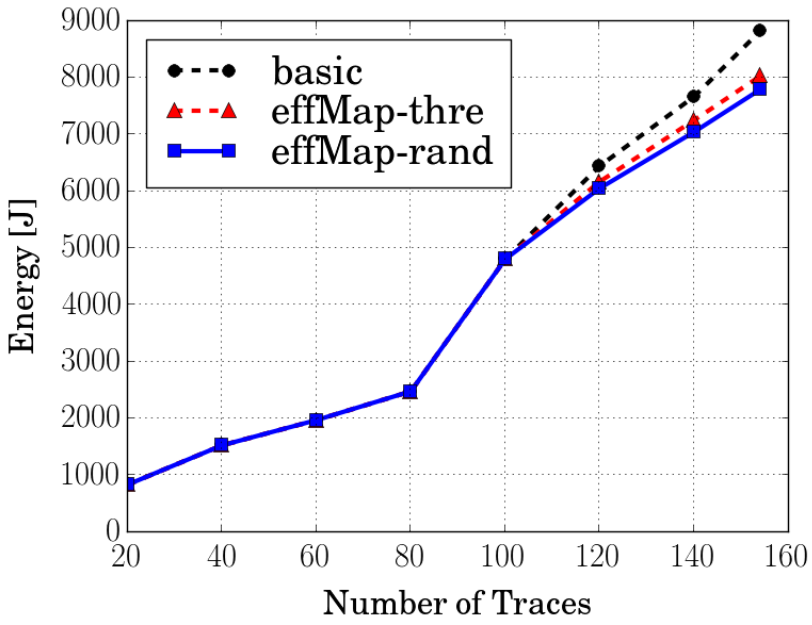


Figure 7.8: Total energy consumption of all devices.

proach when using efficient position sensing with the probabilistic decision algorithm (see Equation 7.3).

Figure 7.8 shows the energy consumption of the different approaches when running the MapGENIE mapping algorithm, where the x-axis denotes the number of pedestrian traces that are used as input to derive a floor plan. We see that up to 100 traces all approaches consume the same amount of energy. This is the number of traces that MapGENIE needs for setting up an initial floor plan. In this phase efficient position sensing cannot be applied. For more than 100 traces, the derived floor plan already contains a precise representation of the complete hallway skeleton (see [PBD+14] for details), which can then serve as base for the efficient positioning algorithm. Hence, after more than 100 traces were processed, the efficient positioning algorithm starts to work, since it can now access the hallway layout of the underlying map. Since we are only interested in the performance of the positioning algorithm, the following numbers are relative to the energy that is consumed for 100 traces.

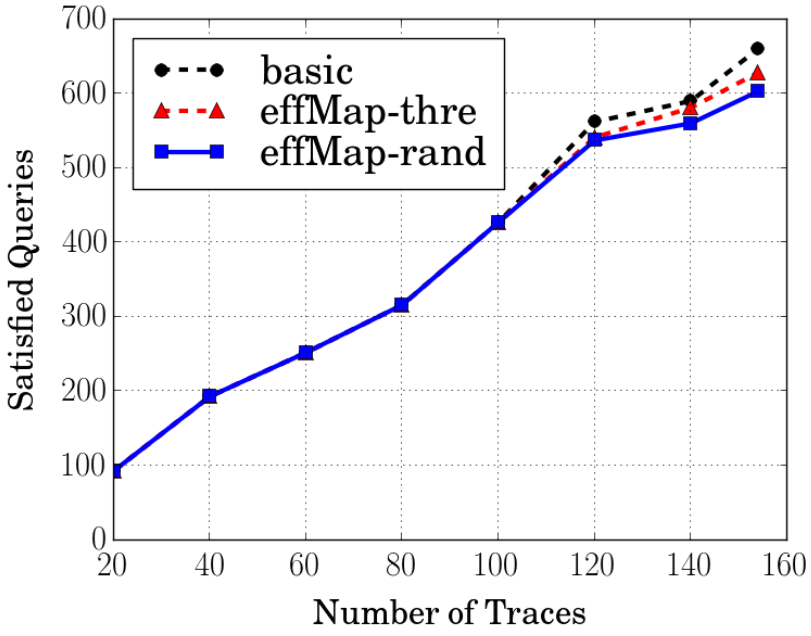


Figure 7.9: Number of satisfied sensing queries.

We see from Figure 7.8 that the probabilistic-approach (effMap-rand) saves in total 25 % of energy compared to the basic approach. For evaluating the threshold-based approach parameter c_p is set to $c_p = 0.5$, which turned out to result in the best results during the evaluations. The approach achieves energy savings of 20 % in comparison to the basic approach and, thus, is not as efficient as the probabilistic approach, but still outperforms the basic approach.

7.5.4 Sensing Effectiveness

Next, we have a look on the impact of the optimized approaches regarding the sensing effectiveness. For this purpose, we look on the number of satisfied sensing queries. Figure 7.9 shows that the optimized approaches cannot satisfy as much sensing queries as the basic approach. More precisely, the probabilistic-approach reaches 76 % of the queries of the basic approach, while the threshold-based approach satisfies 80 % of queries.

We see that the loss in sensing effectiveness is almost as high as the gain in energy savings. Unlike the other approaches that were presented in this thesis,

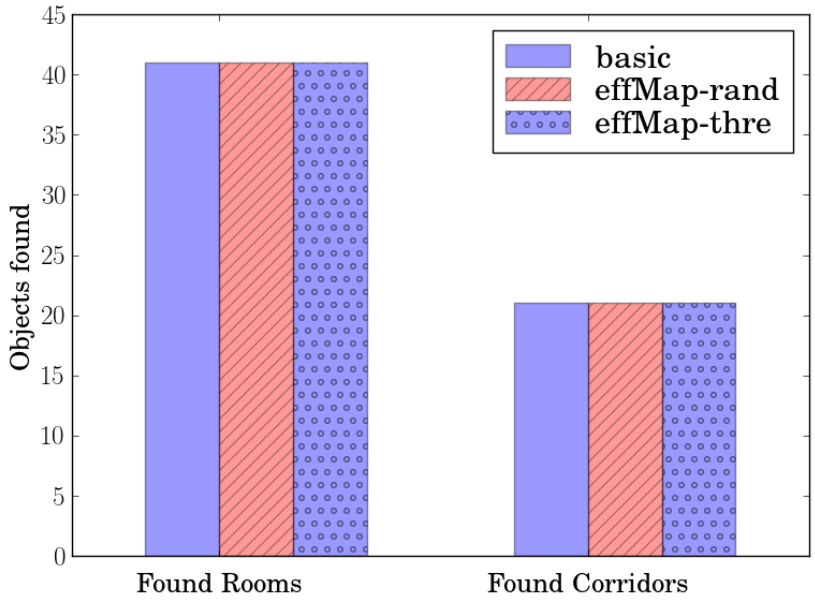


Figure 7.10: Rooms and corridors that are present in the derived floor plan.

the benefits of increased energy efficiency come in this approach with a loss in sensing effectiveness. However, for some applications the reduction of sensing effectiveness may not be critical since the application that utilizes the sensor data is robust against such a small decrease in quality of the sensed data. To show such a case, we have a look on how this reduction in sensing effectiveness influences the outcome of the MapGENIE system. Hence, we compare the floor plan generated by the basic approach and the plan generated by the optimized approaches against the ground truth floor plan. Figure 7.10 shows how many of the rooms and corridors from the ground truth floor plan were found in the respective approaches. More precisely, we compare the positions of rooms and corridors in the generated plans to the ground truth plan and check how many of them are at the same position. We see that when using efficient position sensing, in both approaches the same number of rooms and corridors are found as using the basic approach. Hence, even though the sensing effectiveness declined, we can see that MapGENIE is robust against this decrease and still provides the same quality of the generated floor plan, while saving energy for positioning.

7.6 Related Work

So far, there is no related work that explicitly focuses on improving energy efficiency of indoor position sensing in PS. However, there exist different approaches that propose the combination of WiFi fingerprinting and an IPS to enhance indoor positioning. Noh et al. [NYL+13] propose an infrastructure-free positioning system using dead reckoning and WiFi beacons sent by peer devices for positioning. Other work [WSE+12; RKM13; SCZ+13] uses WiFi fingerprints to correct the error of drift-based IPS traces. They identify landmark spots within the building with the help of WiFi scans and then correct the direction drift of the IPS traces to match the landmark. The efficient position sensing that is presented in this chapter uses a similar approach but for a different goal. Instead of correcting a drift-based trace, it uses WiFi to (re-)start the IPS without any available absolute position.

Other approaches for indoor position sensing use Bluetooth beacons [HNS03; Oks14] or pre-deployed RFID-Tags [ZWX+13]. However, the presented approach does not rely on environments being prepared with specialized equipment, but instead exploits the already available deployments of WiFi access points.

7.7 Conclusion

This chapter presented an approach that reduces positioning energy for indoor PS scenarios. This is achieved by avoiding the energy-intensive IPS with the help of WiFi anchor points that can be collected energy efficiently on-the-fly. To control the switch between these two positioning methods, a positioning scheduler was introduced. This scheduler adaptively activates and deactivates the IPS based on the current set of sensing queries and a device mobility model. The evaluation shows that this approach can save up to 25 % energy. Although the sensing effectiveness is also declining in this case, we saw that applications that are robust to such a decrease in quality can save energy without resulting in a decreased quality of the output model.

8 Summary and Future Work

As a conclusion, this chapter summarizes the contributions of this thesis and discusses possible future extensions.

8.1 Summary

This thesis advances the state of the art in PS research by presenting concepts that increase the energy efficiency of PS systems on mobile devices. To this end, it addressed two of the most energy-intensive operations that are part of a PS system, namely query distribution and device positioning. In the following, each contribution is summarized in detail.

As a first contribution, an opportunistic position update protocol was developed. The goal of this protocol is to provide the query distribution with position information of mobile devices. To keep the energy overhead of this protocol as small as possible, a new kind of update protocol was introduced that is based on so-called opportunistic updates. The idea behind these opportunistic updates is to send position updates together with regular traffic of a mobile device, since this reduces the energy overhead for sending the updates significantly. The evaluation of this approach showed that this protocol lowers the energy consumption for sending position updates by up to 70 % with respect to existing position update protocols.

The second contribution of this thesis is a novel approach that efficiently distributes sensing queries from the PS server to mobile devices. This approach utilizes the information provided by the opportunistic position update protocol to identify a set of devices that are promising candidates to capture data for a sensing query. In contrast to existing approaches, which distribute queries to all devices, the presented approach reduces communication energy since only

a small set of devices are receiving a query. To implement this approach, a probabilistic sensing model was developed that quantifies the likelihood that a device successfully reads data for a query. With the help of this model, a small receiver set of mobile devices can be selected for each query that promises high sensing success. The evaluations showed that this approach reduces the energy consumption on mobile devices by up to 50 % without compromising the sensing effectiveness.

The third contribution of this thesis increases the energy efficiency of outdoor position sensing. Most existing PS approaches assume that a mobile device continuously runs a positioning system like GPS to determine where it is currently located. This is in general necessary to determine when a device should read sensor data. Facing the high energy consumption of continuous positioning, this thesis introduced an approach that reduces energy overhead for positioning without affecting the sensing effectiveness of the PS system. For this purpose, the concept of adaptive positioning was introduced, that implements the idea to temporarily turn off the energy consuming positioning sensor, when a device is far from any sensing range. In outdoor scenarios this is implemented, by calculating an optimal time frame for which the next position fix can be delayed without missing a sensing range. The evaluation showed that this approach reduces the energy for GPS positioning by up to 90 % in comparison to continuous GPS positioning.

The last contribution of this thesis extends the concept of efficient position sensing to indoor scenarios. Since GPS is not available indoors, a mobile device relies on an IPS for indoor positioning. To apply adaptive position sensing also when using an IPS, an efficient indoor positioning approach was proposed that allows to temporarily disable the energy-intensive IPS when moving indoors. To be able to turn the IPS on before reaching a sensing range, a WiFi position recovery system was introduced that is based on WiFi anchor points. These anchor points provide absolute positions, which can be used as starting points for the IPS. With the help of this concept, the positioning energy for indoor PS scenarios can be decreased by up to 25 % while also having to accept an

according decrease in sensing effectiveness of the system.

8.2 Future Work

The work presented in this thesis could be extended in different directions by including additional aspects in the sensing system. In the following, we have a look at three possible extensions that build upon the contributions of this thesis.

One possible extension would be the use of more advanced query semantics for the PS system. In this thesis, a sensing query is a one-time request for sensor data from a certain geographic location. An example for advanced query semantics is for instance a query semantic that includes periodical aspects of data reading. More precisely, instead of requesting data only once, a client could request data in certain time intervals, e.g., having a sensor reading at a certain place at least once every minute. This can for instance be useful to conduct a long-time study on a certain phenomenon, e.g., measuring the fluctuation of the noise level at a certain point over a day. To implement these query semantics, the efficient query distribution needs to be extended to also consider a longer temporal horizon. For this purpose, a new sensing prediction model needs to be designed that not only considers passing a query range once, but also multiple times in the future.

Another interesting research direction is the adaption of the presented concepts to also consider fairness in terms of energy consumption between mobile devices. The optimization criterion that is followed in this thesis is to minimize the total energy consumption over all devices. However, this does not imply an equal energy overhead for different devices. For instance, it is possible that a single device may receive a large number of sensing queries, even though the total energy consumption is very small over all devices. Hence, one further dimension for optimizing is to evenly distribute the energy overhead between devices. One approach towards achieving such energy fairness could be to also consider the device energy when distributing sensing queries. For instance, a device that has already received a large number of queries in the past is less likely to receive

further sensing queries in the future. To implement this idea, the predictive sensing model of a device could be supplemented by a predictive energy model. On the basis of this model and the past energy consumption of a device, the query distribution could be adapted to yield for fairness.

Considering mobile device positioning, one further extension is the use of multiple positioning systems. In this thesis, GPS was assumed for outdoor positioning, since it is currently the most accurate positioning system. Furthermore, it does not require any additional reference data, which is for instance necessary for WiFi or cellular positioning [PKG10]. However, if we assume that such reference data is available and other positioning systems can be used, the adaptive positioning algorithm could be extended. More precisely, the adaptive positioning could not only decide on delaying a position fix but also determine which positioning system should be used for the next position fix. This decision could be driven by the accuracy and energy characteristics that the available positioning methods provide. For instance, if no accurate position is currently required, the system could fall back to a less accurate and more energy-efficient positioning mode. An optimal algorithm would schedule the operation of the different positioning systems in such a way that the total positioning energy is minimized while the sensing effectiveness of the PS system is preserved. First approaches that propose the use of multiple positioning systems have already been proposed [LKL+10; CTS+14]. However, they are designed for a very general scenario and need to be customized for the use in a PS system.

List of Figures

List of Tables

List of Algorithms

Publications

[BDR12a] P. Baier, F. Dürr, and K. Rothermel. „PSense: Reducing Energy Consumption in Public Sensing Systems.“ In: *Proceedings of the Conference on Advanced Information Networking and Applications (AINA)*. 2012, pp. 136–143.

[BDR12b] P. Baier, F. Dürr, and K. Rothermel. „TOMP: Opportunistic traffic offloading using movement predictions.“ In: *Proceedings of the Conference on Local Computer Networks (LCN)*. 2012, pp. 50–58.

[BDR13a] P. Baier, F. Dürr, and K. Rothermel. „Efficient Distribution of Sensing Queries in Public Sensing Systems.“ In: *Proceedings of the Conference on Mobile Ad-Hoc and Sensor Systems (MASS)*. 2013, pp. 272–280.

[BDR13b] P. Baier, F. Dürr, and K. Rothermel. „Opportunistic Position Update Protocols for Mobile Devices.“ In: *Proceedings of the Conference on Pervasive and Ubiquitous Computing (UbiComp)*. 2013, pp. 787–796.

[BPD+14] P. Baier, D. Philipp, F. Dürr, and K. Rothermel. „Quality-based Adaptive Positioning for Energy-Efficient Indoor Mapping.“ In: *Technical Report – Institute for Parallel and Distributed Systems, Universität Stuttgart*. 2014.

[BWD+11] P. Baier, H. Weinschrott, F. Dürr, and K. Rothermel. „MapCorrect: Automatic Correction and Validation of Road Maps Using Public Sensing.“ In: *Proceedings of the Conference on Local Computer Networks (LCN)*. 2011, pp. 58–66.

[BWD11] P. Baier, H. Weinschrott, and F. Dürr. „Effiziente automatisierte Erstellung von Straßenkarten.“ In: *7. GI/ITG KuVS-Fachgespräch. Ortsbezogene Anwendungen und Dienste.* 2011.

[PBD+14] D. Philipp, P. Baier, C. Dibak, F. Dürr, K. Rothermel, S. Becker, M. Peter, and D. Fritsch. „MapGENIE: Grammar-enhanced Indoor Map Construction from Crowd-sourced Data.“ In: *Proceedings of the Conference on Pervasive Computing and Communications (PerCom).* 2014, pp. 139–147.

Bibliography

[AAB+07] T. Abdelzaher, Y. Anokwa, P. Boda, J. Burke, D. Estrin, L. Guibas, A. Kansal, S. Madden, and J. Reich. „Mobiscopes for Human Spaces.“ In: *IEEE Pervasive Computing* 6 (2007), pp. 20–29.

[ABC+13] V. Agarwal, N. Banerjee, D. Chakraborty, and S. Mittal. „USense – A Smartphone Middleware for Community Sensing.“ In: *Proceedings of the Conference on Mobile Data Management (MDM)*. 2013, pp. 56–65.

[AK04] J. N. Al-Karaki and A. E. Kamal. „Routing techniques in wireless sensor networks: a survey.“ In: *IEEE Wireless Communications* 11 (2004), pp. 6–28.

[AK14] H. Amintoosi and S. Kanhere. „A Reputation Framework for Social Participatory Sensing Systems.“ In: *Mobile Networks and Applications* 19 (2014), pp. 88–100.

[AKS+13] A. Albers, I. Krontiris, N. Sonehara, and I. Echizen. „Coupons as Monetary Incentives in Participatory Sensing.“ In: *Collaborative, Trusted and Privacy-Aware e/m-Services*. Springer Berlin Heidelberg, 2013, pp. 226–237.

[App14] Apple. *iPhone 6 Specifications*. Sept. 2014. URL: `https://www.apple.com/iphone-6/specs/`.

[AR12] M. Angermann and P. Robertson. „FootSLAM: Pedestrian Simultaneous Localization and Mapping Without Exteroceptive Sensors – Hitchhiking on Human Perception and Cognition.“ In: *Proceedings of the IEEE* 100 (2012), pp. 1840–1848.

[ASS+02] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. „Wireless Sensor Networks: A Survey.“ In: *Computer Networks: The International Journal of Computer and Telecommunications Networking* 38 (2002), pp. 393–422.

[BBV09] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani. „Energy Consumption in Mobile Phones: A Measurement Study and Implications for Network Applications.“ In: *Proceedings of the Internet Measurement Conference (IMC)*. 2009, pp. 280–293.

[BDR12a] P. Baier, F. Dürr, and K. Rothermel. „PSense: Reducing Energy Consumption in Public Sensing Systems.“ In: *Proceedings of the Conference on Advanced Information Networking and Applications (AINA)*. 2012, pp. 136–143.

[BDR12b] P. Baier, F. Dürr, and K. Rothermel. „TOMP: Opportunistic traffic offloading using movement predictions.“ In: *Proceedings of the Conference on Local Computer Networks (LCN)*. 2012, pp. 50–58.

[BDR13a] P. Baier, F. Dürr, and K. Rothermel. „Efficient Distribution of Sensing Queries in Public Sensing Systems.“ In: *Proceedings of the Conference on Mobile Ad-Hoc and Sensor Systems (MASS)*. 2013, pp. 272–280.

[BDR13b] P. Baier, F. Dürr, and K. Rothermel. „Opportunistic Position Update Protocols for Mobile Devices.“ In: *Proceedings of the Conference on Pervasive and Ubiquitous Computing (UbiComp)*. 2013, pp. 787–796.

[BEH+06] J. Burke, D. Estrin, M. Hansen, A. Parker, N. Ramanathan, S. Reddy, and M. B. Srivastava. „Participatory sensing.“ In: *Proceedings of the Workshop on World-Sensor-Web (WSW)*. 2006, pp. 117–134.

[Bel57] R. Bellman. „A Markovian Decision Process.“ In: *Indiana University Mathematics Journal* 6 (1957), pp. 679–684.

[BHP04] C. Bettstetter, H. Hartenstein, and X. Perez-Costa. „Stochastic Properties of the Random Waypoint Mobility Model.“ In: *Wireless Networks* 10 (2004), pp. 555–567.

[BPD+14] P. Baier, D. Philipp, F. Dürr, and K. Rothermel. „Quality-based Adaptive Positioning for Energy-Efficient Indoor Mapping.“ In: *Technical Report – Institute for Parallel and Distributed Systems, Universität Stuttgart*. 2014.

[BSC+12] W. Brunette, R. Sodt, R. Chaudhri, M. Goel, M. Falcone, J. Van Orden, and G. Borriello. „Open Data Kit Sensors: A Sensor Integration Framework for Android at the Application-level.“ In: *Proceedings of the Conference on Mobile Systems, Applications, and Services (MobiSys)*. 2012, pp. 351–364.

[BWD+11] P. Baier, H. Weinschrott, F. Dürr, and K. Rothermel. „MapCorrect: Automatic Correction and Validation of Road Maps Using Public Sensing.“ In: *Proceedings of the Conference on Local Computer Networks (LCN)*. 2011, pp. 58–66.

[BWD11] P. Baier, H. Weinschrott, and F. Dürr. „Effiziente automatisierte Erstellung von Straßenkarten.“ In: *7. GI/ITG KuVS-Fachgespräch. Ortsbezogene Anwendungen und Dienste*. 2011.

[Cam94] P. J. Cameron. *Combinatorics: Topics, Techniques, Algorithms*. Cambridge University Press, 1994.

[CBH+04] A. L. Cavilla, G. Baron, T. E. Hart, L. Litty, and E. de Lara. „Simplified simulation models for indoor MANET evaluation are not robust.“ In: *Proceedings of the Conference on Sensor and Ad Hoc Communications and Networks (SECON)*. 2004, pp. 610–620.

[CEL+06] A. T. Campbell, S. B. Eisenman, N. D. Lane, E. Miluzzo, and R. A. Peterson. „People-centric Urban Sensing.“ In: *Proceedings of the Workshop on Wireless Internet (WICON)*. 2006, pp. 2–5.

[CEL+08] A. T. Campbell, S. B. Eisenman, N. D. Lane, E. Miluzzo, R. A. Peterson, H. Lu, X. Zheng, M. Musolesi, K. Fodor, and G. Ahn. „The Rise of People-Centric Sensing.“ In: *IEEE Internet Computing* 12 (2008), pp. 12–21.

[CFB+13] G. Cardone, L. Foschini, P. Bellavista, A. Corradi, C. Borcea, M. Talasila, and R. Curtmola. „Fostering participaction in smart cities: a geo-social crowdsensing platform.“ In: *IEEE Communications Magazine* 51 (2013), pp. 112–119.

[CHH14] M. H. Cheung, F. Hou, and J. Huang. „Participation and reporting in participatory sensing.“ In: *Proceedings of the Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*. 2014, pp. 357–364.

[CHK08] D. Cuff, M. Hansen, and J. Kang. „Urban sensing: out of the woods.“ In: *ACM Communications* 51 (2008), pp. 24–33.

[Cho03] V. Cho. „A comparison of three different approaches to tourist arrival forecasting.“ In: *Tourism Management* 24 (2003), pp. 323–330.

[CLC+13] Z. Chen, M. Lin, F. Chen, N.D. Lane, G. Cardone, R. Wang, T. Li, Y. Chen, T. Choudhury, and A.T. Campbell. „Unobtrusive sleep monitoring using smartphones.“ In: *Proceedings of the Conference on Pervasive Computing Technologies for Healthcare (PervasiveHealth)*. 2013, pp. 145–152.

[CRK+11] D. Christin, A. Reinhardt, S. Kanhere, and M. Hollick. „A survey on privacy in mobile participatory sensing applications.“ In: *Journal of Systems and Software* 84 (2011), pp. 1928–1946.

[CTS+14] Y. Chon, E. Talipov, H. Shin, and H. Cha. „SmartDC: Mobility Prediction-Based Adaptive Duty Cycling for Everyday Location Monitoring.“ In: *IEEE Transactions on Mobile Computing* 13 (2014), pp. 512–525.

[DB12] S. Deng and H. Balakrishnan. „Traffic-aware techniques to reduce 3G/LTE wireless energy consumption.“ In: *Proceedings of the Conference on emerging Networking EXperiments and Technologies (CoNEXT)*. 2012, pp. 181–192.

[DBY+10] J. Dai, X. Bai, Z. Yang, Z. Shen, and D. Xuan. „Mobile Phone-based Pervasive Fall Detection.“ In: *Personal Ubiquitous Computing* 14 (2010), pp. 633–643.

[DMP+10] T. Das, P. Mohan, V. N. Padmanabhan, R. Ramjee, and A. Sharma. „PRISM: Platform for Remote Sensing Using Smartphones.“ In: *Proceedings of the Conference on Mobile Systems, Applications, and Services (MobiSys)*. 2010.

[DS13] E. De Cristofaro and C. Soriente. „Participatory privacy: Enabling privacy in participatory sensing.“ In: *IEEE Network* 27 (2013), pp. 32–36.

[DSJ13] E. D'Hondt, M. Stevens, and A. Jacobs. „Participatory noise mapping works! An evaluation of participatory sensing as an alternative to standard techniques for environmental monitoring.“ In: *Pervasive and Mobile Computing* 9 (2013), pp. 681–694.

[EGH+08] J. Eriksson, L. Girod, B. Hull, R. Newton, S. Madden, and H. Balakrishnan. „The Pothole Patrol: Using a Mobile Sensor Network for Road Surface Monitoring.“ In: *Proceedings of the Conference on Mobile Systems, Applications, and Services (MobiSys)*. 2008, pp. 29–39.

[ELK+08] M. Emiliano, N. D. Lane, F. Kristóf, R. Peterson, H. Lu, M. Musolesi, S. B. Eisenman, X. Zheng, and A. T. Campbell. „Sensing Meets Mobile Social Networks: The Design, Implementation and Evaluation of the CenceMe Application.“ In: *Proceedings of the Conference on Embedded Network Sensor Systems (SenSys)*. 2008, pp. 337–350.

[ESS+11] J. Eisner, A. Spillner, S. Storandt, S. Funke, and A. Herbst. „Algorithms for Matching and Predicting Trajectories.“ In: *Proceedings of the Workshop on Algorithm Engineering and Experiments (ALENEX)*. 2011, pp. 84–95.

[FHR12] S. Föll, K. Herrmann, and K. Rothermel. „Energy-Efficient Update Protocols for Mobile User Context.“ In: *Proceedings of the Conference on Advanced Information Networking and Applications (AINA)*. 2012, pp. 120–127.

[FKS+11] T. Fahrni, M. Kuhn, P. Sommer, R. Wattenhofer, and S. Welten. „Sundroid: Solar Radiation Awareness with Smartphones.“ In: *Proceedings of the Conference on Ubiquitous Computing (UbiComp)*. 2011, pp. 365–374.

[FLR07] T. Farrell, R. Lange, and K. Rothermel. „Energy-efficient Tracking of Mobile Objects with Early Distance-based Reporting.“ In: *Proceedings of the Conference on Mobile and Ubiquitous Systems: Networking and Services (Mobiquitous)*. 2007, pp. 1–8.

[Fox05] E. Foxlin. „Pedestrian tracking with shoe-mounted inertial sensors.“ In: *IEEE Computer Graphics and Applications* 25 (2005), pp. 38–46.

[FRC11] T. Farrell, K. Rothermel, and R. Cheng. „Processing Continuous Range Queries with Spatiotemporal Tolerance.“ In: *IEEE Transactions on Mobile Computing* 10 (2011), pp. 320–334.

[FT87] M. L. Fredman and R. E. Tarjan. „Fibonacci Heaps and Their Uses in Improved Network Optimization Algorithms.“ In: *Journal of the ACM (JACM)* 34 (1987), pp. 596–615.

[GGP14] T. Giannetsos, S. Gisdakis, and P. Papadimitratos. „Trustworthy People-Centric Sensing: Privacy, security and user incentives roadmap.“ In: *Proceedings of the Workshop on Ad Hoc Networking (MED-HOC-NET)*. 2014, pp. 39–46.

[Gmb] Falcom GmbH. *Trace 4 You*. `http://www.trace4you.com/`. URL: `http://get.adobe.com/reader/`.

[HNS03] J. Hallberg, M. Nilsson, and K. Synnes. „Positioning with Bluetooth.“ In: *Proceedings of the Conference on Telecommunications (ICT)*. 2003, pp. 954–958.

[HPI13] S. Hachem, A Pathak, and V. Issarny. „Probabilistic registration for large-scale mobile participatory sensing.“ In: *Proceedings of the Conference on Pervasive Computing and Communications (PerCom)*. 2013, pp. 132–140.

[HSS+12] D. Hasenfratz, O. Saukh, S. Sturzenegger, and L. Thiele. „Participatory air pollution monitoring using smartphones.“ In: *Proceedings of the Workshop on Mobile Sensing: From Smartphones and Wearables to Big Data*. 2012.

[HSW+14] D. Hasenfratz, O. Saukh, C. Walser, C. Hueglin, M. Fierz, and L. Thiele. „Pushing the spatio-temporal resolution limit of urban air pollution maps.“ In: *Proceedings of the Conference on Pervasive Computing and Communications (PerCom)*. 2014, pp. 67–77.

[Inc] Google Inc. *Google Latitude*. `http://www.google.com/latitude/`.

[JLT+11] Y. Jiang, K. Li, L. Tian, R. Piedrahita, X. Yun, O. Mansata, Q. Lv, R. P. Dick, M. Hannigan, and L. Shang. „MAQS: A Personalized Mobile Sensing System for Indoor Air Quality Monitoring.“

In: *Proceedings of the Conference on on Ubiquitous Computing (UbiComp)*. 2011, pp. 271–280.

[JXP+13] Y. Jiang, Y. Xiang, X. Pan, K. Li, Q. Lv, R. P. Dick, L. Shang, and M. Hannigan. „Hallway based automatic indoor floorplan construction using room fingerprints.“ In: *Proceedings of the Conference on Pervasive and Ubiquitous Computing (UbiComp)*. 2013, pp. 315–324.

[KH08] A. Krause and E. Horvitz. „A Utility-theoretic Approach to Privacy and Personalization.“ In: *Proceedings of the Conference on Artificial Intelligence (AAAI)*. 2008, pp. 1181–1188.

[Kjr12] M. B. Kjærgaard. „On Improving the Energy Efficiency and Robustness of Position Tracking for Mobile Devices.“ In: *Mobile and Ubiquitous Systems: Computing, Networking, and Services*. Springer Berlin Heidelberg, 2012, pp. 162–173.

[KKR11] S. Kaiser, M. Khider, and P. Robertson. „A human motion model based on maps for navigation systems.“ In: *EURASIP Journal on Wireless Communications and Networking* 1 (2011), p. 60.

[KLG+09] M. B. Kjærgaard, J. Langdal, T. Godsk, and T. Toftkjær. „EnTracked: Energy-efficient Robust Position Tracking for Mobile Devices.“ In: *Proceedings of the Conference on on Mobile Systems, Applications, and Services (MobiSys)*. 2009, pp. 221–234.

[KOK09] A. Keränen, J. Ott, and T. Kärkkäinen. „The ONE Simulator for DTN Protocol Evaluation.“ In: *Proceedings of the Conference on Simulation Tools and Techniques (SIMUTools)*. 2009.

[KXA+13] W. Z. Khan, Y. Xiang, M. Y. Aalsalem, and Q. Arshad. „Mobile Phone Sensing Systems: A Survey.“ In: *IEEE Communications Surveys Tutorials* 15 (2013), pp. 402–427.

[LC13] Q. Li and G. Cao. „Providing privacy-aware incentives for mobile sensing.“ In: *Proceedings of the Conference on Pervasive Computing and Communications (PerCom)*. 2013, pp. 76–84.

[LCZ+13] N. D. Lane, Y. Chon, L. Zhou, Y. Zhang, F. Li, D. Kim, G. Ding, F. Zhao, and H. Cha. „Piggyback CrowdSensing (PCS): Energy Efficient Crowdsourcing of Mobile Sensor Data by Exploiting Smartphone App Opportunities.“ In: *Proceedings of the Conference on Embedded Networked Sensor Systems (MobiSys)*. 2013.

[LFR+12] H. Lu, D. Frauendorfer, M. Rabbi, M. Mast, G. T. Chittaranjan, A. T. Campbell, D. Gatica-Perez, and T. Choudhury. „StressSense: Detecting Stress in Unconstrained Acoustic Environments Using Smartphones.“ In: *Proceedings of the Conference on on Ubiquitous Computing (UbiComp)*. 2012, pp. 351–360.

[LGY+12] H. Liu, Y. Gan, J. Yang, S. Sidhom, Y. Wang, Y. Chen, and F. Ye. „Push the Limit of WiFi Based Localization for Smartphones.“ In: *Proceedings of the Conference on Mobile Computing and Networking (Mobicom)*. 2012, pp. 305–316.

[LHZ+13] H. Liu, S. Hu, W. Zheng, Z. Xie, S. Wang, P. Hui, and T. Abdelzaher. „Efficient 3G budget utilization in mobile participatory sensing applications.“ In: *Proceedings of the Conference on Computer Communications (INFOCOM)*. 2013, pp. 1411–1419.

[LKL+10] K. Lin, A. Kansal, D. Lymberopoulos, and F. Zhao. „Energy-accuracy Trade-off for Continuous Mobile Device Location.“ In: *Proceedings of the Conference on Mobile Systems, Applications, and Services (MobiSys)*. San Francisco, California, USA, 2010, pp. 285–298. ISBN: 978-1-60558-985-5.

[LLE+10] H. Lu, N. D. Lane, S. B. Eisenman, and A. T. Campbell. „Bubble-sensing: Binding sensing tasks to the physical world.“ In: *Pervasive and Mobile Computing* 6 (2010), pp. 58–71.

[LML+10] N.D. Lane, E. Miluzzo, H. Lu, D. Peebles, T. Choudhury, and A.T. Campbell. „A survey of mobile phone sensing.“ In: *IEEE Communications Magazine* 48 (2010), pp. 140–150.

[LR01] A. Leonhardi and K. Rothermel. „A Comparison of Protocols for Updating Location Information.“ In: *Cluster Computing* 4 (2001), pp. 355–367.

[LWG+09] R. Lange, H. Weinschrott, L. Geiger, A. Blessing, F. Dürr, K. Rothermel, and H. Schütze. „On a generic uncertainty model for position information.“ In: *Proceedings of the Workshop on Quaility of Context (QuaCon)*. 2009, pp. 76–87.

[LYL+10] H. Lu, J. Yang, Z. Liu, N. D. Lane, T. Choudhury, and A. T. Campbell. „The Jigsaw continuous sensing engine for mobile phone applications.“ In: *Proceedings of the Conference on Embedded Networked Sensor Systems (SenSys)*. 2010, pp. 71–84.

[LZD+12] F. Li, C. Zhao, G. Ding, J. Gong, C. Liu, and F. Zhao. „A Reliable and Accurate Indoor Localization Method Using Phone Inertial Sensors.“ In: *Proceedings of the Conference on Ubiquitous Computing (UbiComp)*. 2012, pp. 421–430.

[LZZ11] H. Liu, Y. Zhang, and Y. Zhou. „TailTheft: leveraging the wasted time for saving energy in cellular communications.“ In: *Proceedings of the Workshop on Mobility in the Evolving Internet Architecture (MobiArch)*. 2011, pp. 31–36.

[Mah84] M. J. Maher. „Estimating the turning flows at a junction: a comparison of three models.“ In: *Traffic Engineering & Control* 25 (1984), pp. 19–22.

[MPF+10] M. Musolesi, M. Piraccini, K. Fodor, A. Corradi, and A. T. Campbell. „Supporting Energy-efficient Uploading Strategies for Continuous Sensing Applications on Mobile Phones.“ In: *Proceedings*

of the Conference on Pervasive Computing and Communications (PerCom). 2010, pp. 355–372.

[MPR08] P. Mohan, V. N. Padmanabhan, and R. Ramjee. „Nericell: Rich Monitoring of Road and Traffic Conditions Using Mobile Smartphones.“ In: *Proceedings of the Conference on Embedded Network Sensor Systems (Sensys)*. 2008, pp. 323–336.

[NYL+13] Y. Noh, H. Yamaguchi, U. Lee, P. Vij, J. Joy, and M. Gerla. „CLIPS: Infrastructure-free collaborative indoor positioning scheme for time-critical team operations.“ In: *Proceedings of the Conference on Pervasive Computing and Communications (PerCom)*. 2013.

[Oks14] I. Oksar. „A Bluetooth signal strength based indoor localization method.“ In: *Proceedings of the Conference on Systems, Signals and Image Processing (IWSSIP)*. 2014, pp. 251–254.

[OSM] OSM. *Open Street Maps*. `http://www.openstreetmap.org/`.

[Ott89] W. Ott. „Human activity patterns: a review of the literature for estimating time spent indoors, outdoors, and in transit.“ In: *Proceedings of the Research Planning Conference on Human Activity Patterns*. 1989.

[PAG+06] T. Pering, Y. Agarwal, R. Gupta, and R. Want. „CoolSpots: reducing the power consumption of wireless mobile devices with multiple radio interfaces.“ In: *Proceedings of the Conference on Mobile Systems, Applications, and Services (MobiSys)*. 2006, pp. 220–232.

[PBD+14] D. Philipp, P. Baier, C. Dibak, F. Dürr, K. Rothermel, S. Becker, M. Peter, and D. Fritsch. „MapGENIE: Grammar-enhanced Indoor Map Construction from Crowd-sourced Data.“ In: *Proceedings of the Conference on Pervasive Computing and Communications (PerCom)*. 2014, pp. 139–147.

[PDR11] D. Philipp, F. Dürr, and K. Rothermel. „A Sensor Network Abstraction for Flexible Public Sensing Systems.“ In: *Proceedings of the Conference on Mobile Ad-hoc and Sensor Systems (MASS)*. 2011, pp. 460–469.

[PHZ12] A. Pathak, Y. V. Hu, and M. Zhang. „Where is the Energy Spent Inside My App?: Fine Grained Energy Accounting on Smartphones with Eprof.“ In: *Proceedings of the Conference on Computer Systems (EuroSys)*. 2012, pp. 29–42.

[PJH+12] A. Pathak, A. Jindal, Y. C. Hu, and S. P. Midkiff. „What is Keeping My Phone Awake?: Characterizing and Detecting No-sleep Energy Bugs in Smartphone Apps.“ In: *Proceedings of the Conference on Mobile Systems, Applications, and Services (MobiSys)*. 2012, pp. 267–280.

[PKG10] J. Paek, J. Kim, and R. Govindan. „Energy-efficient Rate-adaptive GPS-based Positioning for Smartphones.“ In: *Proceedings of the Conference on Mobile Systems, Applications, and Services (MobiSys)*. 2010, pp. 299–314.

[PLL10] B. Priyantha, D. Lymberopoulos, and J. Liu. „Enabling Energy Efficient Continuous Sensing on Mobile Phones with LittleRock.“ In: *Proceedings of the Conference on Information Processing in Sensor Networks (IPSN)*. 2010, pp. 420–421.

[PMR14] C.G. Pendao, AC. Moreira, and H. Rodrigues. „Energy consumption in personal mobile devices sensing applications.“ In: *Proceedings of the Conference on Wireless and Mobile Networking Conference (WMNC)*. 2014, pp. 1–8.

[PSA+13] D. Philipp, J. Stachowiak, P. Alt, F. Durr, and K. Rothermel. „DrOPS: Model-driven optimization for Public Sensing systems.“ In: *Proceedings of the Conference on Pervasive Computing and Communications (PerCom)*. 2013, pp. 185–192.

[QWG+10a] F. Qian, Z. Wang, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck. „Characterizing radio resource allocation for 3G networks.“ In: *Proceedings of the Internet Measurement Conference (IMC)*. 2010, pp. 137–150.

[QWG+10b] F. Qian, Z. Wang, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck. „TOP: Tail Optimization Protocol For Cellular Radio Resource Allocation.“ In: *Proceedings of the Conference on Network Protocols (ICNP)*. 2010, pp. 285–294.

[RCK+10] R. K. Rana, C. T. Chou, S. Kanhere, N. Bulusu, and W. Hu. „Ear-phone: an end-to-end participatory urban noise mapping system.“ In: *Proceedings of the Conference on Information Processing in Sensor Networks (IPSN)*. 2010, pp. 105–116.

[REL+14] Kiran K. Rachuri, Christos Efstratiou, Ilias Leontiadis, Cecilia Mascolo, and Peter J. Rentfrow. „Smartphone Sensing Offloading for Efficiently Supporting Social Sensing Applications.“ In: *Pervasive and Mobile Computing* 10 (2014), pp. 3–21.

[RES10] S. Reddy, D. Estrin, and M. Srivastava. „Recruitment framework for participatory sensing data collections.“ In: *Proceedings of the Conference on Pervasive Computing and Communications (PerCom)*. 2010, pp. 138–155.

[RH10] A. Rice and S. Hay. „Decomposing power measurements for mobile devices.“ In: *Proceedings of the Conference on Pervasive Computing and Communications (PerCom)*. 2010, pp. 70–78.

[RHS+06] F. Ricciato, E. Hasenleithner, P. Svoboda, and W. Fleischer. „On the impact of unwanted traffic onto a 3G network.“ In: *Proceedings of the Workshop on Security, Privacy and Trust in Pervasive and Ubiquitous Computing (SecPerU)*. 2006, pp. 8–56.

[RKM13] V. Radu, L. Kriara, and M. K. Marina. „Pazl: A mobile crowdsensing based indoor WiFi monitoring system.“ In: *Proceedings of the Conference on Network and Service Management (CNSM)*. 2013.

[RLR+09] J. Ryder, B. Longstaff, S. Reddy, and D. Estrin. „Ambulation: A Tool for Monitoring Mobility Patterns over Time Using Mobile Phones.“ In: *Proceedings of the Conference on Computational Science and Engineering (CSE)*. 2009, pp. 927–931.

[RM04] K. Romer and F. Mattern. „The Design Space of Wireless Sensor Networks.“ In: *IEEE Wireless Communications* 11 (2004), pp. 54–61.

[RMM+10] K. K. Rachuri, M. Musolesi, C. Mascolo, P. J. Rentfrow, C. Longworth, and A. Aucinas. „EmotionSense: A Mobile Phones Based Adaptive Platform for Experimental Social Psychology Research.“ In: *Proceedings of the Conference on Ubiquitous Computing (UbiComp)*. 2010, pp. 281–290.

[RNL11] Z. Ruan, E. C. Ngai, and J. Liu. „Wireless sensor deployment for collaborative sensing with mobile phones.“ In: *Computer Networks: The International Journal of Computer and Telecommunications Networking* 55 (2011), pp. 3224–3245.

[RPK+12] M. R. Ra, B. Priyantha, A. Kansal, and J. Liu. „Improving Energy Efficiency of Personal Sensing Applications with Heterogeneous Multi-processors.“ In: *Proceedings of the Conference on Ubiquitous Computing (UbiComp)*. 2012, pp. 1–10.

[RPS+10] M. R. Ra, J. Paek, A. B. Sharma, R. Govindan, M. H. Krieger, and M. J. Neely. „Energy-delay tradeoffs in smartphone applications.“ In: *Proceedings of the Conference on Mobile Systems, Applications, and Services (MobiSys)*. 2010, pp. 255–270.

[SCC12] H. Shin, Y. Chon, and H. Cha. „Unsupervised Construction of an Indoor Floor Plan Using a Smartphone.“ In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews* 42 (2012), pp. 889–898.

[SCZ+13] G. Shen, Z. Chen, P. Zhang, T. Moscibroda, and Y. Zhang. „Walkie-Markie: Indoor Pathway Mapping Made Easy.“ In: *Proceedings of the Conference on Networked Systems Design and Implementation (NSDI)*. 2013, pp. 85–98.

[SSM09] A. Shye, B. Scholbrock, and G. Memik. „Into the wild: studying real user activity patterns to guide power optimizations for mobile architectures.“ In: *Proceedings of the Symposium on Microarchitecture (MICRO)*. 2009, pp. 168–178.

[STZ12] X. Sheng, J. Tang, and W. Zhang. „Energy-efficient collaborative sensing with mobile phones.“ In: *Proceedings of the Conference on Computer Communications (INFOCOM)*. 2012, pp. 1916–1924.

[TKT+10] N. Thepvilojanapong, S. Konomi, Y. Tobe, Y. Ohta, M. Iwai, and K. Sezaki. „Opportunistic Collaboration in Participatory Sensing Environments.“ In: *Proceedings of the Conference on Mobility in the Evolving Internet Architecture (MobiArch)*. 2010, pp. 39–44.

[TRL+09] A. Thiagarajan, L. S. Ravindranath, K. LaCurts, S. Toledo, J. Eriksson, S. Madden, and H. Balakrishnan. „VTrack: Accurate, Energy-Aware Traffic Delay Estimation Using Mobile Phones.“ In: *Proceedings of the Conference on Embedded Networked Sensor Systems (SenSys)*. 2009, pp. 85–98.

[TTO+14] T. Tsujimori, N. Thepvilojanapong, Y. Ohta, Y. Zhao, and Y. Tobe. „History-based Incentive for Crowd Sensing.“ In: *Proceedings of the Workshop on Web Intelligence and Smart Sensing (IWWISS)*. 2014, pp. 1–6.

[WDR09] H. Weinschrott, F. Dürr, and K. Rothermel. „Efficient Capturing of Environmental Data with Mobile RFID Readers.“ In: *Proceedings of the Conference on Mobile Data Management (MDM)*. 2009, pp. 41–51.

[WDR10] H. Weinschrott, F. Dürr, and K. Rothermel. „StreamShaper: Coordination algorithms for participatory mobile urban sensing.“ In: *Proceedings of the Conference on Mobile Adhoc and Sensor Systems (MASS)*. 2010, pp. 195–204.

[WSE+12] H. Wang, S. Sen, A. Elgohary, M. Farid, M. Youssef, and R. R. Choudhury. „No Need to War-drive: Unsupervised Indoor Localization.“ In: *Proceedings of the Conference on Mobile Systems, Applications, and Services (MobiSys)*. 2012, pp. 197–210.

[WXX+11] W. Wu, J. Xu, M. Xu, and N. Zheng. „An Enhanced Vector-Based Update Algorithm for Network-Constrained Moving Clients.“ In: *Proceedings of the Conference on Green Computing and Communications (GreenCom)*. 2011, pp. 152–158.

[XEC+07] X. Xiaopeng, H.G. Elmongui, Xiaoyong C., and W.G. Aref. „Place: A Distributed Spatio-Temporal Data Stream Management System for Moving Objects.“ In: *Proceedings of the Conference on Mobile Data Management (MDM)*. 2007, pp. 44–51.

[YXF+12] D. Yang, G. Xue, X. Fang, and J. Tang. „Crowdsourcing to Smartphones: Incentive Mechanism Design for Mobile Phone Sensing.“ In: *Proceedings of the Conference on Mobile Computing and Networking (Mobicom)*. 2012, pp. 173–184.

[Zan09] P. A. Zandbergen. „Accuracy of iPhone Locations: A Comparison of Assisted GPS, WiFi and Cellular Positioning.“ In: *Transactions in GIS* 13 (2009), pp. 5–25.

[ZWX+13] H. Zou, H. Wang, L. Xie, and Q. Jia. „An RFID indoor positioning system by using weighted path loss and extreme learning machine.“ In: *Proceedings of the Conference on Cyber-Physical Systems, Networks, and Applications (CPSNA)*. 2013, pp. 66–71.

[ZWZ+13] F. Zhang, D. Wilkie, Y. Zheng, and X. Xie. „Sensing the Pulse of Urban Refueling Behavior.“ In: *Proceedings of the Conference on Pervasive and Ubiquitous Computing (UbiComp)*. 2013, pp. 13–22.

[ZZL12] P. Zhou, Y. Zheng, and M. Li. „How Long to Wait?: Predicting Bus Arrival Time with Mobile Phone Based Participatory Sensing.“ In: *Proceedings of the Conference on Mobile Systems, Applications, and Services (MobiSys)*. 2012, pp. 379–392.

Erklärung

Ich erkläre hiermit, dass ich, abgesehen von den ausdrücklich bezeichneten Hilfsmitteln und den Ratschlägen von jeweils namentlich aufgeführten Personen, die Dissertation selbstständig verfasst habe.

(Patrick Baier)