Efficient Range-Free Monte Carlo Localization for Mobile Wireless Sensor Networks

Dissertation

zur Erlangung des mathematisch-naturwissenschaftlichen Doktorgrades
Doctor rerum naturalium (Dr. rer. nat.)
der Georg-August-Universität zu Göttingen

im PhD Programme in Computer Science (PCS)
der Georg-August University School of Science (GAUSS)

vorgelegt von

Salke Hartung
geboren in Bad Hersfeld

Göttingen
im Oktober 2015

Betreuungsausschuss:	Prof. Dr. rer. nat. Dieter Hogrefe, Georg-August Universität Göttingen Prof. Dr. rer. nat. Xiaoming Fu, Georg-August Universität Göttingen
Referent:	Prof. Dr. rer. nat. Dieter Hogrefe, Georg-August Universität Göttingen
Koreferenten:	Prof. Dr. rer. nat. Xiaoming Fu, Georg-August Universität Göttingen Prof. Dr. rer. nat. Stephan Sigg, Aalto University Helsinki, Finland
Weitere Mitglieder der Prüfungskommission:	Prof. Dr. rer. nat. Carsten Damm, Georg-August Universität Göttingen Prof. Dr. rer. nat. Konrad Rieck, Georg-August Universität Göttingen Prof. Dr. rer. nat. Stephan Waack, Georg-August Universität Göttingen

Tag der mündlichen Prüfung: 20. November 2015

Abstract

A major problem of localization algorithms for Wireless Sensor Networks (WSNs) is the dependence on anchor information. In mobile networks, single node or even whole parts of the network might be temporarily isolated from all anchors, which will render further localization impossible. Consequently, the localization error increases drastically in these situations. In addition to that, network operators have a strong interest in reducing the number of costly anchor nodes to reduce deployment and operation costs, which further amplifies the problem of missing anchor information.

This thesis first discusses the advantages and disadvantages of two large groups of localization algorithms, range-based and range-free localization, and provides a study of an often used ranging technique based on *Received Signal Strength Indicator* (RSSI) distance estimation. After that, two new variants of a well-known range-free localization approach called *Monte Carlo Localization* (MCL) are introduced to account for the problem of missing or temporarily unavailable seed nodes. In *Sensor-Assisted Monte Carlo Localization* (SA-MCL) a dead reckoning technique using additional information provided by magnetometer and accelerometer sensors is proposed to update the location estimation based on the last known position. *Path-Oriented Monte Carlo Localization* (PO-MCL) is designed to exploit path-based mobility behavior as it exists for several applications in WSNs to improve the localization process. Both approaches are evaluated using excessive network simulation. Furthermore, SA-MCL is implemented on real hardware and evaluated in a mobile WSN testbed formed by radio controlled cars. It is shown that in low seed density situations the localization error for SA-MCL and PO-MCL can be reduced by about 60% and 50%, respectively.

Zusammenfassung

Das Hauptproblem von Lokalisierungsalgorithmen für WSNs basierend auf Ankerknoten ist die Abhängigkeit von diesen. Mobilität im Netzwerk kann zu Topologien führen, in denen einzelne Knoten oder ganze Teile des Netzwerks temporär von allen Ankerknoten isoliert werden. In diesen Fällen ist keine weitere Lokalisierung möglich. Dies wirkt sich primär auf den Lokalisierungsfehler aus, der in diesen Fällen stark ansteigt. Des weiteren haben Betreiber von Sensornetzwerken Interesse daran, die Anzahl der kosten- und wartungsintensiveren Ankerknoten auf ein Minimum zu reduzieren. Dies verstärkt zusätzlich das Problem von nicht verfügbaren Ankerknoten während des Netzwerkbetriebs.

In dieser Arbeit werden zunächst die Vor- und Nachteile der beiden großen Hauptkategorien von Lokalisierungsalgorithmen (range-based und range-free Verfahren) diskutiert und eine Studie eines oft für range-based Lokalisierung genutzten Distanzbestimmungsverfahren mit Hilfe des RSSI vorgestellt. Danach werden zwei neue Varianten für ein bekanntes range-free Lokalisierungsverfahren mit Namen MCL eingeführt. Beide haben zum Ziel das Problem der temporär nicht verfügbaren Ankerknoten zu lösen, bedienen sich dabei aber unterschiedlicher Mittel. SA-MCL nutzt ein dead reckoning Verfahren, um die Positionsschätzung vom letzten bekannten Standort weiter zu führen. Dies geschieht mit Hilfe von zusätzlichen Sensorinformationen, die von einem elektronischen Kompass und einem Beschleunigungsmesser zur Verfügung gestellt werden. PO-MCL hingegen nutzt das Mobilitätsverhalten von einigen Anwendungen in Sensornetzwerken aus, bei denen sich alle Knoten primär auf einer festen Anzahl von Pfaden bewegen, um den Lokalisierungsprozess zu verbessern. Beide Methoden werden durch detaillierte Netzwerksimulationen evaluiert. Im Fall von SA-MCL wird außerdem eine Implementierung auf echter Hardware vorgestellt und eine Feldstudie in einem mobilen Sensornetzwerk durchgeführt. Aus den Ergebnissen ist zu sehen, dass der Lokalisierungsfehler in Situationen mit niedriger Ankerknotendichte im Fall von SA-MCL um bis zu 60% reduziert werden kann, beziehungsweise um bis zu 50% im Fall von PO-MCL.

Acknowledgements

I would like to thank my supervisors Prof. Dieter Hogrefe and Prof. Xiaoming Fu for their support and pieces of good advice during my conducted studies. I am expressing my gratitude to Prof. Stephan Sigg, who kindly agreed to be my third thesis supervisor for the dissertation procedure.

I am grateful for the inspirational ideas provided by Prof. Konrad Rieck which led to some of the outcomes presented in this thesis.

I am obliged to Prof. Carsten Damm and Prof. Stephan Waack for being further members of the examination committee.

Finally, I would like to thank my family, colleagues and friends, who always offered a friendly ear to listen to my concerns and who accompanied my progress with motivating words.

The work in this thesis is developed in the context of the project
Securing Communications in Internet of Things (IoT) Environments
funded by the Simulationswissenschaftliches Zentrum Clausthal-Goettingen.

SWZ Clausthal – Göttingen
Simulationswissenschaftliches Zentrum

Contents

List of Figures

List of Tables

Listings

Acronyms

a/d *analog/digital*

APIT *Adapted Point-in-Triangle*

API *Application Programming Interface*

AoA *Angle of Arrival*

BDS *BeiDou Navigation Satellite System*

COT *Center of Gravity*

DSN *Distributed Sensor Networks*

DARPA *Defense Advanced Research Projects Agency*

DMP *Digital Motion Processor*

DSSS *Direct Sequence Spread Spectrum*

EU *European Union*

ESA *European Space Agency*

ETT *Expected Transmission Time*

FHHS *Frequency Hopping Spread Spectrum*

GPS *Global Positioning System*

GNSS *Global Navigation Satellite Systems*

GSM *Groupe Spécial Mobile*

GUI *Graphical User Interface*

LAN *Local Area Network*

IoT *Internet of Things*

los *line of sight*

MA-MCL *Mobility-Assisted Monte Carlo Localization*

MCL *Monte Carlo Localization*

MCB *Monte Carlo localization boxed*

MMCL *Multihop-Based MCL*

nesC *network embedded systems C*

NMEA *National Marine Electronics Association*

ots *off the shelf*

PCI *Peripheral Component Interconnect*

PIT *Point-in-Triangle*

PO-MCL *Path-Oriented Monte Carlo Localization*

RSSI *Received Signal Strength Indicator*

RTT *Round-Trip Time*

SA-MCL *Sensor-Assisted Monte Carlo Localization*

SDF *Software Defined Radio*

TDoA *Time Difference of Arrival*

ToA *Time of Arrival*

UMTS *Universal Mobile Telecommunications System*

UTM *Universal Transverse Mercator*

WAN *Wide Area Network*

WMCL *Weighted MCL*

WLAN *Wireless Local Area Network*

WSN *Wireless Sensor Network*

Chapter 1

Introduction

A *Wireless Sensor Network* (WSN) is a network formed by small-scale computing devices equipped with a set of hardware sensors. Facing various restrictions in computational power and energy resources [1, 2, 3], developers of WSNs have to put special attention to the efficiency of all algorithms in the network including data collection, routing and data aggregation.

WSNs are used to collect data in a variety of applications and to transfer that data to a central instance, where it is further analyzed, processed and archived [4]. Without a direct association of the collected data to the spatial information (i.e., where the data has been collected) most of the information is rendered useless. For example, in an agricultural sensor network collecting information about soil humidity on a field it makes no sense to collect just the plain values of the humidity sensors. Instead, the information has to be linked to each sensor's position to infer the condition of the soil. In other applications the sensor network might be rapidly deployed or be of mobile nature. This entails the final position of the sensors cannot be determined a priori. Examples of these networks are sensors thrown out of an airplane to monitor and analyze forest fires [5] and volcano activity [6] or sensors set adrift on the sea to analyze water pollution or ocean currents [7]. Furthermore, other algorithms in WSNs often rely on geographic information, e.g., geographic routing [8, 9, 10] or data aggregation from certain regions of the network [11, 12, 13].

In summary, data collected in a WSN usually needs to be linked to the geographic position of the sensor node and the position information is often used for other algorithms. The action of a node to determine its own location is called the localization process. A trivial possibility to achieve this goal, which is also suitable for mobile nodes, is using the *Global Positioning System* (GPS). However, there exist various issues with this approach including high energy consumption, large antenna size and high deployment costs, which render it unsuitable for the extensive usage in WSNs [14, 15]. As a consequence, alternative approaches have to be investigated. Over the last two decades of research in sensor networks, the concept of anchor nodes and location announcements emerged [16, 17, 18, 19, 20, 21]. Anchor nodes are always aware of their position, either by having a fixed location or by being equipped with GPS [22]. They send out location announcements to assist simple nodes in estimating their position. Simple nodes collect these location announcements and calculate an estimate for their own position using a localization algorithm. The aim in localization is to keep the number of necessary anchor nodes as low as possible, while also maintaining a low localization error.

While the localization process is less complicated for static networks, in which the nodes are not supposed to move after the deployment phase, considerable expenditure is required for mobile sensor networks. Mobile WSNs often have applications in the biology sector. For instance, wildlife monitoring is a prominent interdisciplinary field of research, which can be enhanced in many ways using sensor network technology including tracking single or groups of animals. Prominent examples of deployed wildlife WSNs aim to monitor the position of migratory birds or penguin colonies [23, 24] using localization algorithms to gather insights about the natural behavior of these animals. Advances in the manufacturing size of sensor modules nowadays even allow much smaller creatures to be equipped with sensors [25, 26]. In other applications, sensor networks might be planted in oceans [27] or lakes to determine pollution levels or to explore ocean currents.

All of the mentioned examples share the fact that the sensor nodes are mobile. Mobile WSNs have to face several additional challenges, e.g., changing network topologies, network splits, and isolated nodes. Consequently, positioning information must be constantly updated for each node.

Localization algorithms can be roughly divided into range-free and range-based solutions [28, 17]. While the former rely on receiving location announcements only, the latter additionally employ active sensing to gather further information like distances to other nodes or the angle of the incoming signal. Past and ongoing research mainly focuses on range-based approaches. One reason for this trend are the manifold possibilities of processing data collected by ranging techniques, which emerges in a broad spectrum of localization algorithms as shown in [29, 30, 31, 32]. However, almost all ranging techniques are subject to several problems including additional costs, inaccuracy or increased computational overhead [33, 34, 35]. As a consequence this thesis advocates the facilitation of range-free algorithms for localization.

1.1 Problem Statement

A popular representative of range-free localization is *Monte Carlo Localization* (MCL) [36]. MCL uses a set of samples to estimate a node's location where each sample represents a possible location of the node. Using anchor information, impossible locations are filtered. Due to its simplicity and robustness, MCL is an attractive solution for the localization process in sensor networks. However, just like any other algorithm based on anchor node information, a node will be unable to update its position estimation if it loses contact to all anchor nodes. Consequently, if the node is moving, the localization error will heavily increase in these situations. The central research question of this thesis is how temporary scenarios in which no anchor information is available can be bypassed, while maintaining a reasonable low localization error. Furthermore, network operators have a strong interest in using as less as possible anchor nodes, since they are more expensive and of high-maintenance nature. Therefore, this thesis also explores possibilities to reduce the number of anchor nodes in the network to maintain a certain localization error.

1.2 Thesis Main Contributions

In this thesis an initial study of the most common technique used in range-based localization is performed. In theory, the *Received Signal Strength Indicator* (RSSI) allows estimating distances based on the power of the received signal. However, one key finding of this study is that relying on distance estimation based on signal strength leads to very unsteady results and is not suitable for the application in rapidly deployed or mobile sensor networks. The conducted study motivates the focus on range-free solutions in the following parts of the thesis.

After that, two solutions to solve the problem of missing anchor information for the MCL algorithm are proposed. In SA-MCL additional sensor information from common sensors like magnetometer and accelerometer is used to determine the path a node travels relative to its last known position. The approach is initially evaluated using network simulation. After the simulations, a field test study in a mobile WSN testbed is performed. The testbed uses radio controlled cars to introduce mobility in the network. Studies on real hardware are very rare, especially evaluations performed in mobile WSNs [37]. SA-MCL can reduce the localization error by up to 60 % when compared to MCL.

The second approach additionally focuses on reducing the number of extra sensors required in SA-MCL for applications with a certain mobility behavior. Often, nodes in a WSN will move on a finite set of paths. PO-MCL tries to exploit this behavior by mapping the paths to a grid structure held in memory. In situations without anchor information the grid acts as a reference to predict the movement of the node. PO-MCL is evaluated using professional network simulation software. The results indicate that the localization error can be reduced by up to 50 % when compared to MCL.

In short, the contributions of this thesis can be summarized in the following way:

- Review of well-known ranging techniques and state of the art range-free localization algorithms for WSNs.
- Analysis of the suitability of RSSI as a distance estimator for the usage in range-based localization algorithms in WSNs.

- Design and implementation of two new variants of the well-known MCL localization method named SA-MCL and PO-MCL to reduce the localization error and save expenses for costly anchor nodes.
- Analysis of both approaches using network simulation tools.
- Design and deployment of a mobile WSN testbed based on radio controlled cars to evaluate SA-MCL in a real environment.

1.3 Thesis Impact

This section gives a full list of all publications as well as supervised theses and projects.

First author publications:

- **Sensor-Assisted Monte Carlo Localization for Wireless Sensor Networks. Salke Hartung**, Somayeh Taheri, and Dieter Hogrefe. In 6th IEEE International Conference on Cyber Technology (CYBER), Hong Kong, HK, June 2014.
- **Sensor-Assisted Monte Carlo Localization for Wireless Sensor Networks. Salke Hartung**, Ansgar Kellner, Arne Bochem, and Dieter Hogrefe. In 6th IFIP International Conference on New Technologies, Mobility and Security (NTMS) - Poster + Demo Session, Dubai, UAE, April 2014.
- **Practical RSSI Long Distance Measurement Evaluation in Wireless Sensor Networks. Salke Hartung**, Henrik Brosenne, and Dieter Hogrefe. In The 2013 IEEE Conference on Wireless Sensors (ICWiSe 2013), Kuching, Malaysia, December 2013.

Co-author publications:

- **Anonymous Group-Based Routing in MANETs.** Somayeh Taheri, **Salke Hartung**, and Dieter Hogrefe. In Journal of Information Security and Applications. Elsevier, 2014.
- **Anonymity and Privacy in Multicast Mobile Ad Hoc Networks.** Somayeh Taheri, **Salke Hartung**, and Dieter Hogrefe. In The 6th ACM International Conference on Security of Information and Networks (ACM/SIGSAC SIN 2013), Aksaray, Turkey, November 2013.

- **RDIS: Destination Location Privacy in MANETs.** Somayeh Taheri, **Salke Hartung**, and Dieter Hogrefe. International Journal of Information Privacy, Security and Integrity (IJIPSI), Vol. 1, Nos. 2/3, 2012.
- **Achieving receiver location privacy in Mobile Ad Hoc Networks.** Somayeh Taheri, **Salke Hartung**, and Dieter Hogrefe. In Proceedings of the IEEE International Conference on Information Privacy, Security, Risk and Trust (PASSAT2010), Minneapolis, USA, August 2010.

Supervised Theses and Projects:

- **Efficient Localization for Mobile Wireless Sensor Networks**, Master Thesis, Arne Bochem, 2015
- **An ESRI-Shapefile-based Mobility Model Implementation for the Qualnet Network Simulator**, Bachelor Thesis, Andreas Zdziarstek, 2014
- **Analyse der Sendereichweitebestimmung des Netzwerksimulators Qualnet**, Student project, Andreas Zdziarstek, 2014
- **Entfernungsmessung in Sensornetzwerken.**, "FoLL - Forschungsorientiertes Lehren und Lernen[1] im Sommersemester 2013", University of Goettingen, 2013

1.4 Thesis Organization

This thesis is divided into the following chapters as shown in Figure 1.1.

Chapter 1 is this introduction

Chapter 2 describes the theoretical background of this thesis. It contains necessary explanations, examples and definitions regarding WSNs in general with particular attention paid to the localization process. Further, the notation used in this thesis is presented. Readers with a technical background of WSNs and localization might want to skip this chapter and directly proceed to Chapter 3.

[1]FoLL is an interdisciplinary project established by the University of Goettingen to integrate undergraduate students in research work

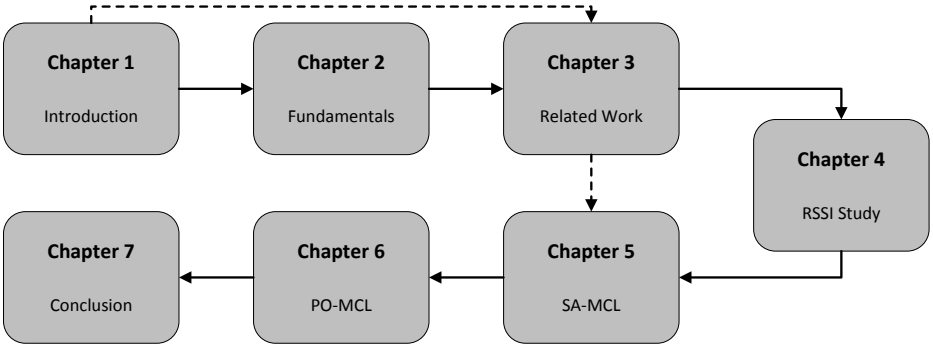


Figure 1.1: Thesis organization.

Chapter 3 summarizes well-known range-free localization approaches and puts special focus on the MCL algorithm, which is the fundamental core this thesis is built on top on.

Chapter 4 presents a study considering RSSI as a possible distance estimator for the usage in range-based localization algorithms. The suitability of RSSI is examined in various practical experiments, and sources of impact on RSSI measurements are listed. The findings of the study motivate the use of range-free localization in the remainder of this thesis. Readers familiar with the disadvantages of range-based localization, especially using RSSI, may want to skip this chapter and directly proceed to Chapter 5.

Chapter 5 introduces and evaluates *Sensor-Assisted Monte Carlo Localization* (SA-MCL). After an extensive simulation study, a mobile WSN testbed consisting of radio controlled cars is presented, which is used to evaluate the feasibility of the proposed solution in a real scenario.

Chapter 6 introduces and evaluates *Path-Oriented Monte Carlo Localization* (PO-MCL). Extensive network simulation is performed to evaluate different application scenarios.

Chapter 7 summarizes the conducted research, lists the limitations of this work and gives an outlook on possible future work.

Chapter 2

Prerequisites

In this chapter basic principles of WSNs are revised and fundamentals of localization are explained. Readers with a technical background in WSNs and localization might want to skip this chapter. However, to be able to follow subsequent chapters and to get an understanding of the notation used in this thesis, it is advisable to skim through the terms and definitions introduced in Section 2.2.

Contents

2.1 Wireless Sensor Networks

A WSN is a multihop network formed by very small computer devices, which are highly restricted in their technical capabilities. The main purpose of a WSN is to collect environment data of all kinds. The data is forwarded to a central base station called sink, which usually operates as a gateway to a more sophisticated network responsible for processing the collected data. Details on the limitations and the network architecture of WSNs are given in Section 2.1.2 and Section 2.1.6, respectively. The process of collecting data about an object in the context of a WSN is called sensing. Sensing may include data like temperature values, air pressure, oxygen and other gas levels, soil humidity, positioning data, health related data like blood pressure or cardiac frequency or in short all data, which can be recognized using a hardware sensor. Possible applications and examples of deployed sensor networks are given in Section 2.1.3.

In contrast to wired networks or wireless networks built on the 802.11 standard family [38], WSNs are formed in an ad hoc manner. Instead of directly communicating with a central base station, all participating nodes in the network have to act as forwarding relays. This leads to several constraints and additional challenges demanding highly optimized algorithms fitting the needs of WSNs.

The following paragraphs give detailed information about the historical development of WSNs, list restrictions and limitations of WSNs and provide an overview of popular hardware platforms for sensor nodes. Furthermore, possible applications and example projects are shown.

2.1.1 History of WSNs

Similar to other developments in computer networks, WSNs have their origin in military research. Pretty much as the invention of the Internet, the *Defense Advanced Research Projects Agency* (DARPA) was heavily involved in conceiving the concept of hardware platforms responsible for collecting environment data. After organizing the Distributed Sensor Nets Workshop in 1978 (DAR 1978) DARPA also founded the *Distributed Sensor Networks* (DSN)

project during the early 1980s [1, 39]. While DARPA mainly focused on (military) applications, later research targeted the design of a unified hardware platform, which was not designed for one application in particular, but could act as a universal sensing platform. As a main contributor to this development, the University of Berkely came up with the Smart Dust project [40] which focused on the design of very small hardware platforms they named motes. Up to today the Berkeley mote is considered to be the conceptual pioneer and technical prototype of all following generations of sensor motes developed by other manufacturers. For a long time WSNs were considered to be an entirely own field of research. With the upcoming development of the *Internet of Things* (IoT), WSNs are considered to be a possible application of the IoT.

2.1.2 Constraints in WSNs

Due to its architecture and design principles a WSN faces several restrictions and constraints [1]:

- **Energy**
  A fundamental constraint in WSNs is energy. Usually all nodes in a WSN are battery powered or at least battery backed up if another power source is available, for instance, by using solar cells. As a direct consequence, the WSN node and software architecture completely focuses on efficient power management. This includes the design of the used hardware, the operating system and all algorithms running the network.

- **Computational Resources**
  Nodes in WSNs have very limited computational capabilities, since they are driven by very low-clocked processors in the range of a few MHz. In addition, the application memory used to store programs and currently processed data is very small and the size of only a few kilobytes.

- **Transmission Range**
  One of the most energy wasting operations in a WSN is network communication, in particular the transmission of packets. As a result and

to keep the power consumption as low as possible, nodes in a WSN have a very limited transmission range. Although the theoretical transmission range often exceeds hundreds of meters, the effective transmission range is remarkably lower.

- **Wireless Operation Mode**
  As WSNs only operate in wireless mode, they are prone to the same challenges as in 802.11 networks (like signal attenuation, reflection, fractioning, etc.) with additional constraints resulting from the limited hardware capabilities.

- **Autonomous Operation Mode**
  A WSN is designed to operate completely on its own. Depending on the application scenario a WSN must be able to bootstrap completely unattended after deployment and must be able to reconfigure in case of topology changes, broken links, physical damage of single nodes, etc.

2.1.3 Application Areas of WSNs

Application areas of WSNs are manifold. The following is a list of some examples, but raises no claim to completeness.

- **Environment Monitoring**
  A classic application for WSNs is the monitoring of all kinds of data related to the environment [41, 42, 43]. This can include very obvious things like weather parameters (temperature, air pressure, wind speed), but is also used for surveillance of volcanos, flood and earthquake regions or pollution levels close to factories and power plants. In agriculture scenarios farmers can monitor the status of their fields, identify areas which need more watering or fertilization or check the filling levels of feeding troughs and waterholes.

- **Exploration of Rivers and Oceans**
  Robust and waterproof sensor motes can be thrown into rivers and oceans to monitor the water quality or to explore streams [44, 45, 46]. These networks are considered to be mobile as the flow of water will

change the position of the nodes over time. There are also scenarios for underwater applications to monitor methane sources, geologic activities or to study the behavior of schools of fish.

- **Smart Cities**
  Smart Cities is the summarizing term for city management aspects in which sensor data is used to react fast and efficiently to certain needs. A typical example is street light management. Nowadays solutions usually switch on all lights at programmed points in time or are controlled by a single light sensor. In Smart Cities street lights can be turned off and on depending on the ambient light levels. Another example is waste management. Intelligent litter boxes can record their fill level and provide this information to garbage collection companies. Using fill level information optimal garbage collection routes can be planned and hotspots of littering can be identified. Building Smart Cities is a central aspect of future city planning strategies and gets lots of attention in research [47, 48, 49, 50].

- **Disaster Management**
  In the event of natural or other disasters a quickly deployed WSN might be the only option to get an overview of the disaster area. Existing infrastructure might be damaged or destroyed or simply not available in the disaster area [51, 52, 53].

- **Wildlife Tracking/Monitoring**
  To gather information about the behavior of animals, WSNs can be used to record health parameters, dynamic behavior of herds or traveling routes of migrating birds [54, 55]. For wildlife tracking animals have to carry a sensor mote fitted to the special challenges in this scenario. The hardware has to be very robust against weather influences and physical damage while not interfering with the animals behavior. A very prominent example of a deployed WSN for wildlife monitoring is the CraneTracker project (see Section 2.1.4).

- **Home Automation**
  More recently WSNs gathered lots of attention in the sector of home automation [56, 57, 58, 59]. This mainly focuses on monitoring parameters of the own residence like heating, lights, motion detectors, surveil-

lance cameras, etc. However, in home automation additional functionality is required. Users like to interact with their infrastructure and change parameters of things which are connected to the network (e.g., change temperature of heating, switch on/off lights). Nevertheless the home automation is an important example of deployed micro WSNs and of course also a prominent example of the Internet of Things.

- **Military Applications**
  Although manifold applications in the military sector can be thought of [60, 61, 62], this part is only mentioned for the sake of completeness and will not be further discussed in this thesis.

2.1.4 Examples of Deployed WSNs

To give an impression how sensor networks find their way into research and industry applications the following lists a few examples of deployed and in use WSNs.

- **CraneTracker Project**
  The CraneTracker project [23, 24] aims to develop an automated method for following and tracking migrating birds such as the endangered whooping crane. In CraneTracker a sensor mote is constructed specially designed for capturing movement data and positions which is important for analyzing bird behavior. The system is equipped with *Groupe Spécial Mobile* (GSM) for long distance data transmission, GPS for capturing location information, accelerometer and digital compass to record movement and resting phases, and a solar panel to recharge batteries. All data is sent to a processing backend which gives a visual presentation of all information.

- **Siega System**
  The Siega System is a specialized commercial platform by Libelium [63] used for monitoring agriculture factors in wine yards or greenhouses. It is a complete system of minor scale (10-50 sensor nodes) including a backend for processing and presenting data. Deployed sensors collect weather information and soil condition as well as hu-

midity and leaf wetness. The collected data can then be used to create optimal conditions for the raised plants.

- **EU-China Dialogue on Smart Cities**
  Formally started in 2013 the EU-China Dialogue on Smart Cities [64] project explored the suitability of several smart city aspects in deployed WSNs. Including some of the biggest cities in Europe and China (e.g., Barcelona, Copenhagen, Frankfurt, ..., Beijing, Shanghai, ...) and funded by the EU and Chinese government the EU-China Dialogue on Smart Cities is one of the largest WSN projects ever conducted.

- **WINSOC project**
  The Wireless Sensor Networks with Self-Organization Capabilities for Critical and Emergency Applications (WINSOC) [65, 66] project is funded by the EU commission and a cooperation between research facilities in Europe and India. The main goal is to build fail-safe and self-healing networks which can operate even in the event of natural disasters. Special focus lies on the early detection of land slides which are likely to appear in the deployment area around Munnar, Idukki District, state of Kerala, South India. The system sent out a warning in 2009 of a possible impending land slide and proved its functionality.

- **Off-The-Shelve Products**
  With the help of commercial enterprises focusing on manufacturing *off the shelf* (ots) components, tiny-scaled WSNs find their way into sectors like farming, agriculture and home automation. Companies like Libelium or Advanticsys provide several case studies in which their products are shown to be a helpful addition for monitoring the status of cattle and greenhouses. The systems often feature complete solutions from modular sensor components, which can be combined in an easy way, through to web front ends and applications to present the collected data to the user.

2.1.5 Hardware Components in WSNs

As already mentioned in Section 2.1.1, a single entity in a WSN is referred to as *sensor mote* or short only *mote*. In another terminology with higher regard

to networking a sensor mote might also be called a single node of the WSN. In this thesis the terms mote, sensor mote and node are used equivalently.

The typical architecture of a WSN node can be divided in 3 subsystems [1], which are connected via communication buses, respectively a standardized interface such as the Serial Peripherical Interface (SPI), the general purpose input/output (GPIO) or the inter-integrated circuit (I^2C).

2.1.5.1 Sensing Subsystem

The sensing subsystem includes all hardware sensors and necessary circuits for *analog/digital* (a/d) converters. It provides an interface for communication with the processing subsystem, which can request sensor data in a polling manner. New data is sent by the sensing system to the processing subsystem whenever it is requested. Most sensors provide analogue data, i.e., different voltage levels corresponding to the measured values. An a/d converter is required to provide quantized values for further processing of the data. A sensor combined with an a/d converter supporting a ready to use interface is called a sensor unit.

2.1.5.2 Processing Subsystem

The processing subsystem includes the main processor and the memory of the device. All programs including the operating system are stored in a non-volatile flash memory. Runtime data such as program variables and recently collected sensor data is stored in the program memory. In addition, a node might provide a special data memory which is used to store sensor data for longer periods of time. Often, this data memory is multiple times larger compared to the program memory. It is worth to note that the processing subsystem is not following the design of the von-Neumann architecture, but the Harvard architecture [67], since it is using separated memory for program instructions and data.

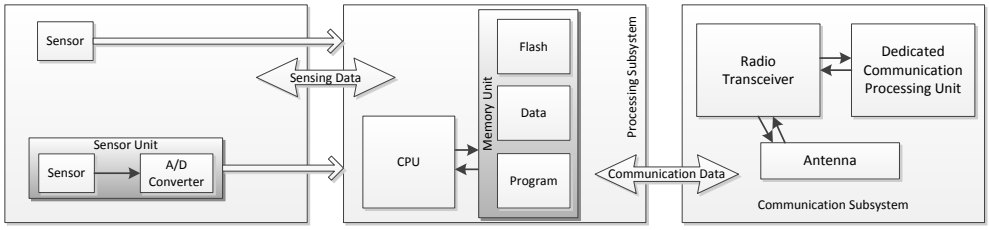


Figure 2.1: Sensormote schema.

2.1.5.3 Communication Subsystem

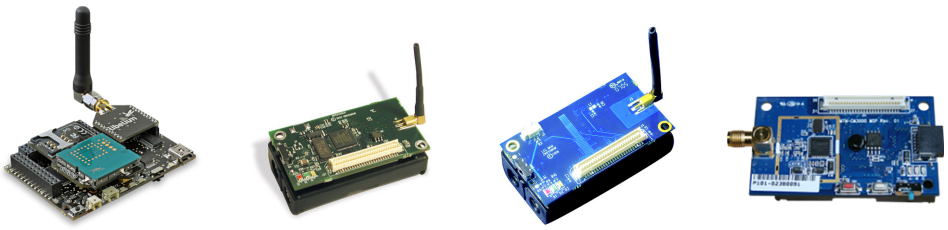
Sending and Receiving packets is controlled by the communication subsystem. This includes the antenna and the radio transceiver (transmitter/receiver) chip as well as an interface to the processing subsystem. A special processing unit for modulation/demodulation tasks might be present if this is not handled by the radio chip itself.

A graphical overview of the components of a sensor mote platform is shown in Figure 2.1. Although this schematic layout looks very modular, all of the 3 subsystems are usually placed on a single board.

2.1.5.4 Popular Sensor Mote Platforms

The following provides a short overview of popular sensor platforms used in research and industrial applications. To a greater or lesser extent all platforms have the same hardware capabilities of a few MHz processing power and about 4-8 KB of RAM. However, depending on the exact components important differences can occur. Therefore, at the beginning of a research project it is important to chose suitable components.

- **WASP Mote by Libelium**
  The WASP Mote [68] is a ready-to-use product targeting deployable applications. It is built for robust outdoor tasks and to survive even under extreme weather conditions. The WASP mote is built as a modular hardware system which means it can be easily adapted to meet the application's requirements by equipping it with different radio modules (e.g., GSM, ZigBee, RFID) and different sensor boards. The main



(a) WASP Mote (b) MicaZ Mote (c) IRIS Mote (d) CM3000 Mote

Figure 2.2: Examples of sensor mote platforms.

board is currently driven by a 14 MHz ATmega1281 processor and 8 KB of RAM. Besides the pure hardware components Libelium offers a complete backend solution with cloud support and automated database storage of sensor data.

- **MicaZ Mote by Crossbow Technologies**
  Equipped with an ATmega128L processor the MicaZ mote [69] features 8 MHz processor power and 4 KB of RAM. The communication unit of the MicaZ mote (MPR2400) follows the 802.15.4 standard and supports a transmission rate of 250 kbps and AES encrypted transmission. To log data, the mote is equipped with a dedicated 512 KB log memory. Crossbow Technologies designed this sensor platform as a universal device which can be connected to any kind of sensor. Therefore, it provides interfaces for several bus communication standards including analog inputs, digital i/o, I^2C, SPI and UART interfaces. Crossbow additionally provides preconfigured sensor boards which are connected via the 51-pin expansion connector.

- **IRIS Mote by Crossbow Technologies**
  The IRIS motes [70] are almost identical compared to MicaZ motes except for the double sized RAM of 8 KB and a different transceiver module (RF230 Atmel).

- **CM3000 Mote by Advanticsys**
  The CM3000 sensor mote [71] by Advanticsys are adapting the original Berkeley mote and are based on the open source TelosB mote [72]. In contrast to the above models, the Advanticsys motes are powered by an 8 MHz Texas Instruments processor (TI MSP430F1611 [73]) and

(a) Fractus BlueTooth Transceiver (b) Coin sized Mote

Figure 2.3: Small scale WSN components.

transceiver module (TI CC2420 [74]). Another difference is the lack of a fixed antenna. Instead, arbitrary external antennas can be connected, which allows to utilize antennas with higher gain or even directional antennas.

All mentioned sensor platforms are shown in Figure 2.2. The listed items are built for general purpose research usage. Components designed for specialized applications can be built much smaller and will offer interesting future applications. Figure 2.3 shows two examples of very small components which can be used in a WSN. The left image shows a chip antenna combining Bluetooth, ZigBee and WLAN built by Fractus [75] which is smaller than a rice corn and therefore can be used to build extremely small devices. The right image shows the prototype of a very compact sensing platform developed by the Japanese company NMEMS Technology Research Organization. The whole sensing module fits on a 7mm $\times$ 7mm square platform and reaches only coin size when equipped with the smallest antenna and powered by a coin cell [76].

2.1.6 Network Architecture of WSNs

WSNs are designed as multi-hop networks which means that in contrast to a *Wireless Local Area Network* (WLAN) as defined by the 802.11 standard family packets are not directly transmitted to the intended receiver in a single-hop manner (see Figure 2.4), but forwarded hop by hop. In addition, there is

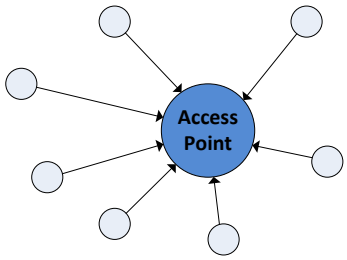


Figure 2.4: Single-Hop architecture in traditional WLANs.

no central entity, e.g., a router, which is responsible for organizing the communication, i.e., set up routes, control data flow in the network, exclude over-excessive nodes, etc. As a consequence, the complete networking management process is organized in a decentralized manner in WSNs. Every node is responsible for finding a route to its intended communication partner and for forwarding packets if it is part of another route.

The major part of the communication can be assigned to forwarding the data collected by the sensors. As mentioned above, all data in a WSN is typically propagated towards a central network entity called the *sink*. The sink acts as a gateway to a higher level network or is equipped with a data storage system to save the collected data for a longer period of time until it is collected by a network maintainer. Routing in WSNs is done on the fly, i.e., there are no predefined gateways or routes during deployment. The design of routing protocols for WSNs is an enormous part of research conducted in WSNs and lots of protocols have been proposed with regard to different aspects and routing metrics important for WSNs such as energy levels, hop count or *Expected Transmission Time* (ETT).

The multi-hop design of a WSN is illustrated in Figure 2.5. The figure shows the sink as the central entity and a bunch of nodes with communication radius r. If a node has no direct neighbor as a communication partner it is excluded from the network communication and considered as *isolated* from the network as shown in the upper right corner. It is also possible that the network is temporarily split into two or more parts. Furthermore, it can be seen that a possible route to the sink may not necessarily be the shortest one (with regard to hop count), as the routing algorithm might consider things like energy levels or number of neighbors and chose forwarding nodes accordingly.

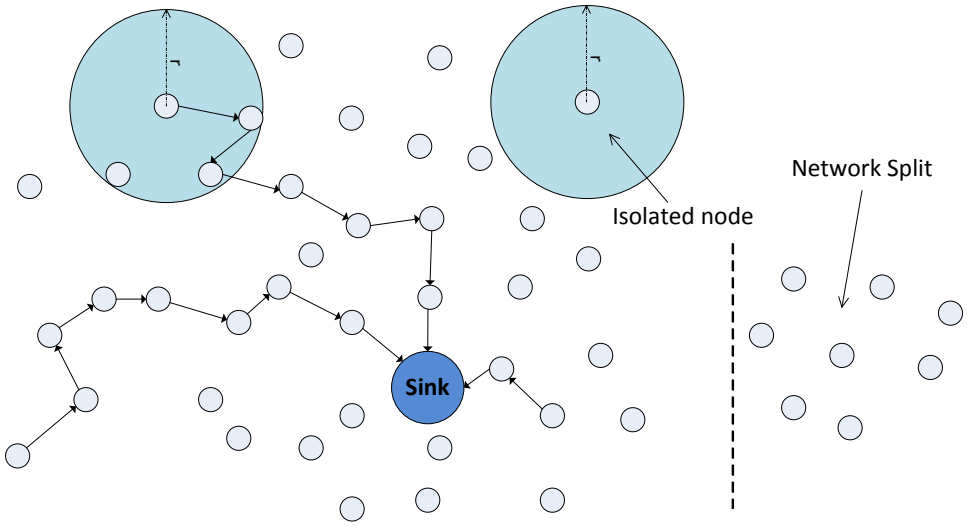


Figure 2.5: Multi-hop architecture in WSNs.

2.1.7 Node Mobility

WSNs can differ in their way of mobile behavior. Depending on the application nodes can be static, semi-mobile if only parts of the network are able to move, or fully mobile if all nodes are able to move in the deployment area.

2.1.7.1 Static WSNs

The most obvious type of a WSN with regard to mobility is no mobility at all, i.e., all nodes are static and will never move during runtime on their own. Static WSNs are usually only configured once during deployment time. Changes in the network topology, communication routes and node neighborhood are only expected if a node in the network completely fails. A typical example for a static WSN is monitoring agriculture where nodes could be deployed in a grid topology on a cornfield.

2.1.7.2 Rapid Deployment

Rapid deployed WSNs emerge when sensor motes are quickly placed in their field of operation without prior choosing their exact final destination.

Examples are motes thrown out of an airplane to gather information on forest fires or polluted areas. In rapid deployment situations the network has to deal with the emerging random network topology and some nodes might even get lost completely, because they are isolated from communication. Often, motes are not expected to be collected after their mission, since they are considered to be damaged (e.g., due to heat, water, etc.) or the risk of retrieving them is too high. Rapid deployed sensor networks must be able to completely bootstrap and configure on their own without external control.

2.1.7.3 Mobile WSNs

In mobile WSNs all nodes are assumed to be moving in the deployment area. The type of movement might differ depending on the application type the most general assumption is to allow all nodes to move arbitrary. Complete mobility leads to several new challenges for the operation of the network. Established communication routes might fail due to broken links and must be reestablished. Single nodes might completely lose contact to the network. In this time they will not be able to forward their collected data and have to buffer it until a connection to the network is reestablished. Depending on the size of the data buffer, some data might have to be dropped.

2.1.7.4 Semi-Mobile WSNs

In a semi-mobile WSN some nodes are able to move or all nodes are able to move at certain times. A possible application might be robots traveling on a deployment site which are acting as relays to forward data from otherwise isolated nodes.

2.1.8 Mobility Models

The mobility model in a network describes how a participating node behaves in terms of location, speed and acceleration over time [77]. It is difficult to classify the movement behavior in WSNs, as there are manifold possibilities. However, a few models established due to their general approach of

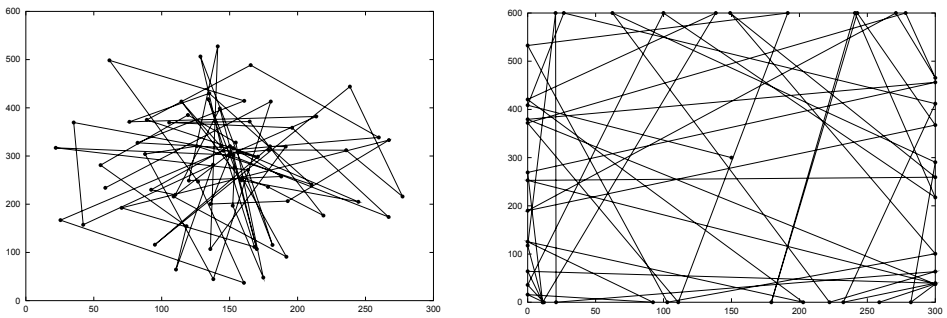
modeling mobility. Mobility models are exceptional important in network simulation software as it has great impact on the quality of the simulations.

2.1.8.1 Random Waypoint Model

The most general case is defined by arbitrary movement. In the random waypoint model [77, 78] a node choses an arbitrary destination in the deployment area and moves to it with a certain velocity. After reaching the waypoint the node will pause for a short random period. The process is repeated until the end of the simulation. Most implementations of the random waypoint model allow defining an interval for the minimum and maximum velocity, $[v_{\min}, v_{\max}]$. For each segment the node will chose a random velocity v with $v_{\min} \leq v \leq v_{\max}$. More detailed implementations also provide simulation of acceleration and deceleration. Since it is very easy to implement and adequately describes arbitrary movement without restrictions, the random waypoint model is one of the most commonly used models for node mobility in scientific work.

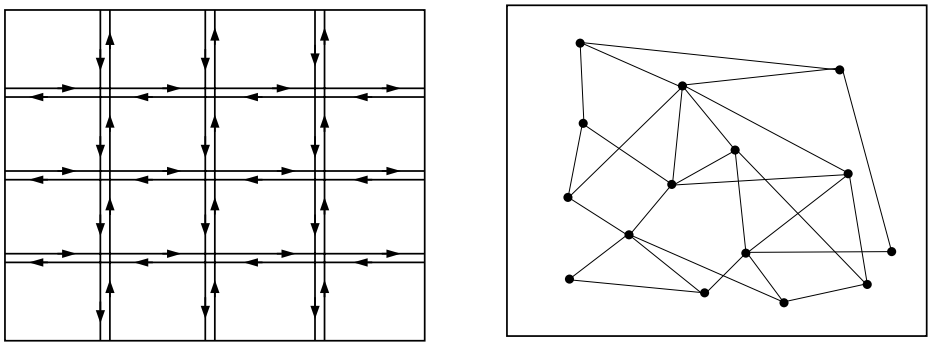
2.1.8.2 Random Walk Model

The random walk model [77, 78] is similar to the random waypoint model, but it uses no pause times and defines a random direction to move in rather than a fixed waypoint to move to. When hitting the deployment area bound-



(a) Random waypoint model (b) Random walk model

Figure 2.6: Illustrations of different mobility models.



(a) Manhattan model. (b) Pathway model

Figure 2.7: Illustrations of graph-based mobility models.

aries the direction is changed depending on the incoming angle. The random walk model was introduced to overcome the clustering problem of the random waypoint model where nodes are often mainly concentrated in the middle of the simulation area instead of using the full extent. Both models are illustrated in Figure 2.6, which is taken from [78].

2.1.8.3 Graph-based Models

Graph-based models are used to model geographic restrictions. This is useful to model areas which cannot be accessed by nodes or to model movement which is only allowed on a set of given paths. A popular representative of a graph-based mobility model is the Manhattan model [77, 78, 79] in which mobility is only allowed on a grid with right-angle intersections. Graph-based models also allow introducing of probabilities for certain segments of the graph and therefore allow modeling of more frequently and less frequently used paths.

A mapping of the random waypoint model to a graph has been proposed by Tian et al. [80]. Instead of randomly choosing a new waypoint, in the pathway model a node is only allowed to chose a random edge of the graph when reaching a vertex. All other parameters are similar to the random waypoint model.

2.2 Localization in Wireless Sensor Networks

As mentioned earlier, collected data in WSNs will only make sense if it is associated with spatial data, i.e., a position ϕ in space. As long as all sensor motes are static, a mapping of a unique sensor ID to its position can be generated offline and does not have to be defined necessarily during deployment time. However, if motes are exchanged or moved to different positions, the mapping would have to be updated every time. Therefore, it is desirable to acquire positioning data (i.e., ϕ) during operation time of the network, especially if some or even all nodes are mobile and can move arbitrarily in the network deployment area.

ϕ can be represented as a vector $\vec{p}$ in a local or global coordinate system. In general, for global coordinates well-known sytems from geographical cartography are used. These systems always relate to a given ellipsoid, which is a mathematical abstraction of planet Earth. The most prominent example is the Gauss/Krueger coordinate system [81] based on the WGS84 [82] ellipsoid. In a local coordinate system $\vec{p}$ can be simply given by $\vec{p} = (x,y,z)$. Using local coordinate systems can be of advantage to simplify position calculations in the network. A mapping to a global coordinate system can easily be done by every node after determining ϕ in the local coordinate system.

Definition 2.1 (Localization in WSNs) *The term localization refers to the process of determining an arbitrary sensor mote's position ϕ in space.*

Definition 2.2 (Self-Localization in WSNs) *The term self-localization refers to the process of a sensor mote determining **its own** position ϕ_{own} in space.*

Both terms are often used equivalent. For the rest of this thesis, if not stated otherwise, the term localization refers to the more precise term self-localization.

Localization must not be confused with the term tracking, which refers to monitoring ϕ of a node over a period of time. Precisely, localization can be used as a tool for tracking applications, but does not have the same meaning.

2.2.1 Usage of Global Navigation Satellite Systems

The easiest way to determine the position ϕ of a mote is to make use of a *Global Navigation Satellite Systems* (GNSS) like GPS. GPS [83] is a nowadays standard and used mainly for navigation systems in cars or mobile phones. GPS was designed by the US military and provides a lower precision service for civil applications. More recently, other systems evolved, mainly the Galileo [84, 14] system by the *European Union* (EU) in cooperation with the *European Space Agency* (ESA), the Russian GLONASS [14, 15] system and the Chinese *BeiDou Navigation Satellite System* (BDS). Although these systems are already in the deployment phase and potentially provide a better spatial resolution, all of them experienced several setbacks including destroyed or malfunctioning satellites [85, 86]. As a consequence it can be assumed that GPS will stay the de facto standard positioning system during the next years. For this reason, the rest of the thesis will refer to the term GPS as a representative for all GNSS systems.

A GPS device directly provides the position vector $\vec{p}$ given as a tuple of coordinates. Additional information includes the current time and level above sea. All information usually is encoded by the *National Marine Electronics Association* (NMEA) [87], which is a plain text format. As a consequence, even if no higher level *Application Programming Interface* (API) is available, the information of the GPS device can be easily parsed.

Although a GPS device provides a very simple solution for determining the own position, it is not suitable for the usage in WSNs due to several reasons.

- **Additional energy consumption**
  Considering the restrictions given in Section 2.1.5, a mote is supposed to save as much energy as possible. However, GPS is known for its comparable high energy consumption and will be a tremendous factor in the energy footprint of a mote.

- **Additional space for chipset and antenna**
  GPS is usually implemented as complete hardware modules which require additional space and an own antenna. Considering some of the application scenarios as described in Section 2.1.3, the mote size is required to be as small as possible.

- **Additional costs**
  In some applications for WSNs like disaster management it is very likely a mote is only used once and will not be recovered after its mission is completed. Especially in these scenarios the costs per mote need to be kept as low as possible. GPS however is a comparable expensive technique.

- **Application scope**
  Many applications in WSNs are either completely designed for indoor operation or target environments where clear line of sight to the sky (i.e., to GPS satellites) might be restricted. GPS requires a clear signal of at least 3 satellites. Considering the fact that GPS signals are often blocked by city skyscrapers or are affected by strong attenuation in forests, GPS is not a feasible localization option.

Because of the given reasons, equipping all motes in a WSN with GPS is not a feasible option. Alternative approaches are necessary to fulfill a sensor mote's requirement of determining its position. The next paragraphs first introduce required terms. After that, the concept of using anchor nodes and localization algorithms is presented.

2.2.2 Basic Terms

Localization in WSNs works with the help of reference points. These can be optical landmarks, which can be recognized by the motes (e.g., using optical recognition systems), but usually some of the participating nodes of the network are operating as reference points themselves. Therefore, they are the only ones equipped with mechanism to determine their position without external help, e.g., by using GPS. In WSNs nodes which act as reference points are called seed nodes. The aim is to keep the amount of costly seed nodes as low as possible.

Definition 2.3 (Seed Nodes in WSNs) *A seed node of a WSN is a node which is always aware of its own location during operation time of the network. Seed nodes act as a set of reference points Ω for others to determine their position.*

Equivalently used terms are anchors and references.

Definition 2.4 (Simple nodes in WSNs) *Nodes which do not have seed node functionality are called simple, ordinary or normal nodes.*

Simple nodes do not have any possibility to determine their position directly. However, they might have additional equipment, which combined with the help of seed nodes, makes it possible to give a position estimation. Seed nodes communicate with simple nodes in an unidirectional nature by sending out location announcements on a regular basis.

Definition 2.5 (Location announcements) *Location announcements are broadcast packets sent by all seed nodes in a regular interval δt. These announcements at least include the position of the seed node and some sort of unique id.*

Definition 2.6 (Observations) *All location announcements received by simple nodes between two localization approaches are treated as new observations, i.e., new information, which helps estimating the own position.*

2.2.3 Localization Algorithms

To avoid the problems of GPS as stated above, with the evolution of WSNs several localization solutions have been developed. Instead of equipping all motes with a dedicated device telling them their positions as GPS does, the task of localization is transferred to the mote itself. With the help of reference points a mote can run a localization algorithm to determine ϕ.

Definition 2.7 (Localization algorithm in WSNs) *A localization algorithm is a procedure executed by every mote in a WSN to estimate its current position ϕ with the help of additional information provided by a set of reference points Ω. The result is subject to a precision error ϵ_{loc}. To clarify the difference between the estimated and the real position of a node the symbols ϕ_{est} and ϕ_{real} are used.*
Input: A set of reference points Ω
Output: ϕ_{est} with error ϵ_{loc}

Every localization algorithm is subject to a precision error called the localization error.

Definition 2.8 (Localization error) *The localization error ϵ_{loc} of a localization algorithm can be calculated as*

$$\epsilon_{loc} = |\phi_{est} - \phi_{real}| \tag{2.2.1}$$

To account for ϵ_{loc} the result of a localization algorithm often is referred to as location estimation, instead of an exactly calculated position. In this thesis the terms location, location estimation, position, position estimation and ϕ_{est} are used equivalently.

2.2.4 Classification of Localization Algorithms

Different approaches exist to classify localization algorithms. Depending on the focus of research this can be classification by mobility constraints of the network, centralized and decentralized approaches, computational constraints, strategy or active/passive operation [16, 17, 28, 88, 89].

A classic approach is to distinguish between *range-based* and *range-free* algorithms [17, 28].

2.2.4.1 Range-based Localization Algorithms

Range-based localization algorithms require possibilities to actively generate some sort of input data from which the position estimation is calculated. This is called the *ranging technique* and can include distance estimations, signal strength calculations, incoming angle determinations or any other form of calculating references relatively to seed nodes.

A very common and trivial example of a range-based algorithm is trilateration [90]. Assume a simple node N_1 wants to find its position ϕ_{est} and 3 seed nodes S_1, S_2, S_3 are available which send location announcements containing

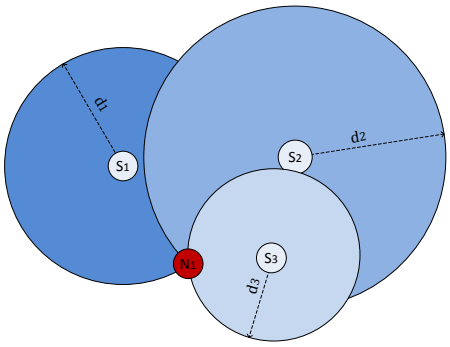


Figure 2.8: Trilateration example.

their position to N_1. Further assume some ranging technique can be used by N_1 to determine the distances d_1, d_2, d_3 to the seed. The emerging circles can be intersected by N_1 to find ϕ_{est} as sketched in Figure 2.8.

Range-based algorithms offer good precision in the optimal case, however, a lot depends on the quality of the ranging technique.

2.2.4.2 Range-free Localization Algorithms

In contrast to range-based approaches, range-free algorithms in general are mainly based on connectivity. They are usually designed to require only minimal computational power and therefore save important resources with regard to processor time, memory and energy. As a consequence, the precision of range-free algorithms cannot compete with the theoretical precision of range-based algorithms. However, the following sections will argue why the performance of range-based algorithms is unsteady and list several other drawbacks of ranging methods.

Table 2.1 summarizes pros and cons of both algorithm classes.

2.2.5 Ranging Techniques

This section gives an overview of popular ranging techniques, discusses advantages and disadvantages of them and lists some applications where ranging is used for localization.

2.2.5.1 Received Signal Strength Indicator (RSSI)

The *Received Signal Strength Indicator* (RSSI) is an integer value which is calculated based on the current signal strength. It is possible to estimate the distance to the communication partner based on the RSSI value, given a suitable path loss model. The path loss model is a mathematical formula used to describe the signal loss, given factors like the distance between sender and receiver, antenna capabilities, transmission power, humidity or walls built of different materials. They are all following the physical fact that signal propagation follows the inverse square law (see Equation (2.2.2)) [91], which means with given transmission power P_t the power of the received signal P_r is not decreasing linearly with distance d. Instead, doubling the distance between two nodes results in a 4-times lower signal strength at the receiver.

$$P_r \propto \frac{P_t}{d^2} \tag{2.2.2}$$

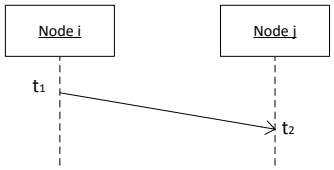
An often used model is the Friis free space propagation model [92, 91] given in Equation (2.2.3). Since receiving power (P_r), transmitting power (P_t), antenna gains at sender and receiver (G_r, G_t) and the wavelength of the signal (λ) are known, it is possible to calculate the distance d between transmitter and receiver by rearranging the equation. The factor L is supposed to account for other losses of all kinds and is often simply set to 1.

$$P_\mathrm{r}(d) = P_\mathrm{t}\frac{G_\mathrm{r}G_\mathrm{t}\lambda^2}{(4\pi d)^2 L} \tag{2.2.3}$$

Propagation models like Friis free space propagation model, ground reflec-

Range-based localization	Range-free localization
+ higher accuracy in optimal case	+ no ranging required
- eventually complex ranging	+ small overhead
- accuracy depends on quality of ranging	+ often easy to implement
- higher algorithm complexity	- lower accuracy

Table 2.1: Pros and cons of range-based and range-free localization.



$$dist_{ij} = (t_2 - t_1) * v \qquad (2.2.4)$$

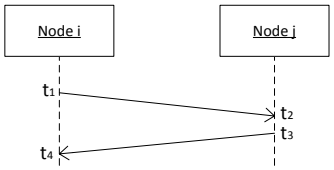
Figure 2.9: ToA distance estimation.

tion 2-way model or log normal shadowing model [91] have not been designed for application in WSNs and are mostly used for calculations in networks with large distances between sender and receiver as they can be found in cellular networks used for mobile telecommunications. For WSNs more specialized path loss models as in [93] and [94] have been developed. However, they all suffer from the common problem that obstacles like houses, trees, rocks, etc. have heavy impact on signal propagation. Nevertheless, RSSI is a very popular ranging technique used in ad hoc networks. Lots of research works using RSSI assume a constant behavior as defined by the path loss model without regard to influences which affect signal propagation. An evaluation study of RSSI as a distance estimator is given in Chapter 4.

2.2.5.2 Time of Arrival (ToA) and Round-Trip Time (RTT)

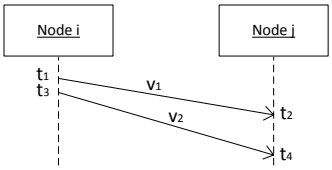
Time of Arrival (ToA) is a method of determining distances based on time stamps. Since radio signals are propagating with the speed of light [91], it is possible to calculate the distance a packet traveled if it is known when the packet has been sent and received. ToA is based on the assumption that radio signals are propagated by the speed of light in a vacuum. This approach requires precisely synchronized clocks at the sender and receiver side. To avoid expensive timing devices at at least one communication end, instead of measuring the one-way ToA, the *Round-Trip Time* (RTT) is usually preferred, i.e., the time from sending a packet until receiving a reply is measured. Both methods are illustrated in Figures 2.9 and 2.10.

ToA is used in the GPS system (see Section 2.2.1) where all GPS satellites are equipped with an atomic clock to avoid a desynchronized system. The GPS consumer clients are usually equipped with simple clocks.



$$dist_{ij} = \frac{(t_4 - t_1) - (t_3 - t_2)}{2} * v \qquad (2.2.5)$$

Figure 2.10: RTT distance estimation.



$$dist_{ij} = (v_1 - v_2) * (t_4 - t_2) \qquad (2.2.6)$$

Figure 2.11: TDoA measurement.

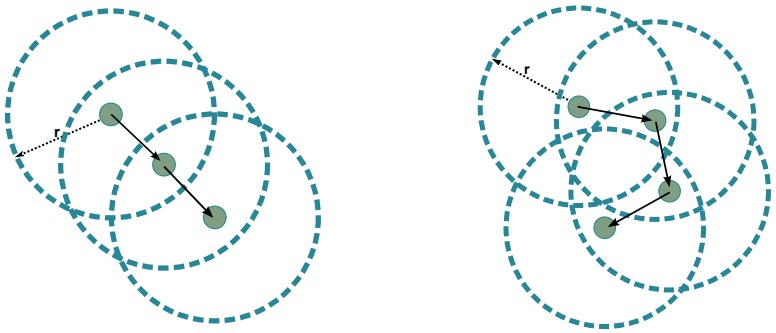
2.2.5.3 Time Difference of Arrival (TDoA)

The *Time Difference of Arrival* (TDoA) approach makes use of two different signals propagated with different speed, e.g., combining radio and audio waves. If sent at the same time, the distance between two nodes i and j can be calculated by taking the different arrival times of both signals into consideration. The process is illustrated in Figure 2.11.

Both ToA and TDoA are used in cellular networks, for instance, for the localization of mobile phones. Although a mobile phone only interacts with one base station for user communication, it is always connected to multiple base stations for planning handovers. Therefore, especially in urban areas where the density of base stations is notably higher, both techniques can be applied.

2.2.5.4 Angle of Arrival (AoA)

It is possible to measure the angle of an incoming signal to determine the direction where the signal must have been propagated. *Angle of Arrival* (AoA) determination can be done for audio waves as well as for radio waves. For audio waves highly sensitive microphones are required which are able to observe minor changes in volume levels depending on the microphone position. AoA with audio waves does not have significant impact in real appli-



(a) Hopcount = 2 (b) Hopcount = 3

Figure 2.12: Distance estimation using hopcounting.

cations because of natural noise which would interfere with measurements. For radio waves angle determination can be accomplished using special antennas which are divided in sectors or arrays of antennas. With three angle estimations the triangulation algorithm [90] can be run. However, AoA measurements either require lots of efforts or expensive hardware and are therefore rarely used in real applications. Even more recent approaches [95] using commercial ots *Software Defined Radio* (SDF) still have a comparatively high price.

2.2.5.5 Hopcounting-based Techniques

Since the maximum transmission range r is usually a known parameter in the network, hop counting can give a rough estimation of the distance between a sending and a receiving node in a WSN. However, as shown in Figure 2.12, this can lead to large errors depending on the network constellation. In the left example of the figure the hop count is 2, i.e., the estimated distance will be set to $2 \times r$, which is roughly correct, because all nodes are almost arranged linear. In the right example the hop count is 3, i.e., the estimated distance will be set to $3 \times r$, although sender and receiver are actually much closer to each other.

2.2.6 Metrics for Evaluating a Localization Algorithm

2.2.6.1 Absolute Localization Error

The most important metric by far for all localization algorithms is precision. Precision in terms of absolute localization error ϵ_{loc} can be defined as the difference in distances from the estimated position ϕ_{est} to the real position ϕ_{real}, as already stated in Definition 2.8 for the localization error ϵ_{loc}. Localization is typically done in $\mathbb{R}^2$ or $\mathbb{R}^3$ so the distance between ϕ_{real} and ϕ_{est} is given by the Euclidean Distance as defined for the two dimensional case in formula 2.2.7.

Definition 2.9 (Localization error calculation) *The localization error ϵ_{loc} of a localization algorithm is determined using the Euclidean Distance*

$$\epsilon_{loc} = \sqrt{(|\phi_{real}.x - \phi_{est}.x|)^2 + (|\phi_{real}.y - \phi_{est}.y|)^2} \tag{2.2.7}$$

In evaluations of range-free localization algorithms the localization error is often given as the quotient of the absolute error and the radio range r as shown in Equation (2.2.8). The quotient is also denoted as $\epsilon_{\text{loc}}(r)$.

Definition 2.10 (Localization error in terms of r)

$$\textit{error-quotient} = \epsilon_{loc}(r) = \frac{\epsilon_{loc}}{r} \tag{2.2.8}$$

2.2.6.2 Computational and Traffic Overhead

The computational overhead describes how expensive it is to determine the position of a node in terms of processor and memory usage. This can be done by counting instruction cycles necessary for running the localization algorithm or by assigning weights to instruction classes. For instance a comparison is executed much faster compared to a multiplication operation in a processor [96]. Traditional complexity analysis can be done to classify an

algorithm based on its input size. However, although many localization algorithms can be found in the same complexity class (e.g., $O(n)$), it is worth having a closer look at performance differences between them. In resource-limited environments like WSNs it is important to eliminate computational overhead wherever possible.

Traffic overhead describes the number of packets necessary to exchange between nodes in order to gather necessary information for the localization algorithm. For instance all seed nodes send location announcements and simple nodes may send packets for ranging purposes. A commonly used metric to describe the traffic overhead of a protocol is the ratio of number of control packets N_{control} and number of data packets N_{data} as shown in Equation (2.2.9) [97].

Definition 2.11 (Traffic overhead)

$$overhead = \frac{N_{control}}{N_{data}} \tag{2.2.9}$$

However, this metric can only be applied if assumptions about the network traffic can be made or if real network traces are available. Therefore, it is usually not used in evaluations of localization algorithms.

2.2.6.3 Power Consumption

One of the most limiting resources in a WSN is energy in terms of battery power. A very important goal of all algorithms in a WSN is to save energy whenever possible. The energy overhead describes how much additional power is required to run an algorithm. Energy overhead is very difficult to capture, since it would require to monitor the processor time and to derive the consumed energy from that. A more convenient way is to directly measure the power consumption of the node and to monitor the battery level over a certain period of time. A common approach is to measure the current draw of a node when running specific algorithms using a multimeter.

2.3 Conception of New Localization Algorithms

2.3.1 Network Simulation

A major problem of designing new algorithms for distributed systems is testing and validation. Deploying new algorithms is a complicated and time consuming process and often not an option at all if a running system would have to be interrupted. A much more convenient solution is network simulation. Depending on the capabilities of the simulation machine, thousands of nodes can be simulated to evaluate scalability and performance of algorithms [98, 99]. A network simulation environment usually consists of a hardware and a software layer. Depending on the sophistication of the simulation the hardware layer can be provided by a single personal computer or a cluster system specialized on multi-threaded tasks. Either way, the software layer on top defines a virtual environment for the network. As a simulation cannot represent the real environment in every detail, all parameters which have impact on an algorithm have to be approximated by a mathematical model [100, 101]. Examples are the path loss models discussed in Section 2.2.5.1 to introduce fading effects or the mobility models discussed in Section 2.1.8 to define a certain mobility behavior of the nodes.

2.3.2 Used Software

Network simulation software is a well-established tool in nowadays computer science research. There are a variety of different simulators available which can be categorized by open-source/free or commercial products, commandline or *Graphical User Interface* (GUI) interface, wired ore wireless networking simulation, etc.

2.3.2.1 MCL Java Simulator

This simulator was originally developed as part of the evaluation of the Monte Carlo Localization algorithm for WSNs [36]. The software was built for only one task and therefore does not provide any other algorithms ex-

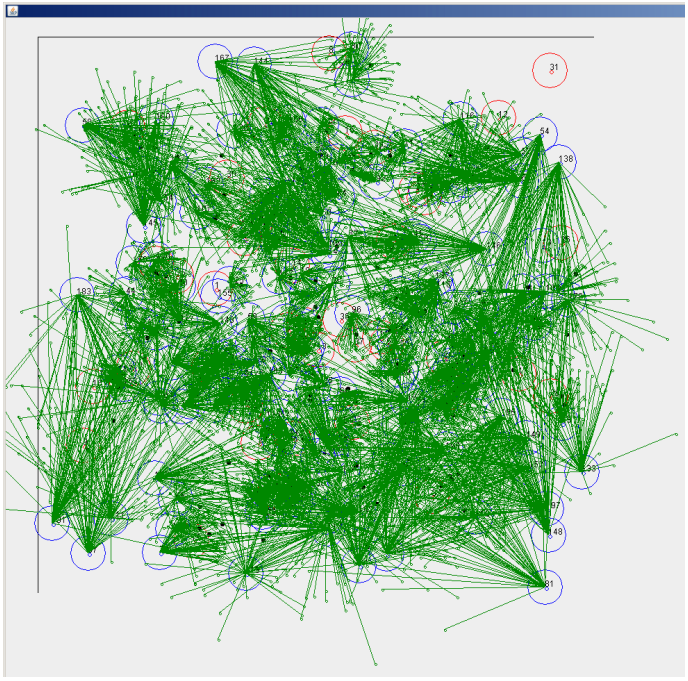


Figure 2.13: Screenshot of custom Java simulator executing a MCL scenario.

cept APIT [102] and Centroid [103], which were implemented for comparison against MCL. Therefore it does not feature an implementation of the ISO/OSI protocol stack [104] and assumes packets can be exchanged in some way if two nodes are in communication range. Although simulations of this kind lack realism, they can be a first indicator for the goodness of an algorithm. In this thesis the simulator is used to validate the quality of the algorithm proposed in Chapter 5. For demonstration purposes a rudimentary GUI was implemented to visualize node behavior and the goodness of the location estimation at any time. A screenshot of the simulator GUI while executing a MCL simulation is shown in Figure 2.13.

2.3.2.2 QualNet

The second simulator used in this thesis is QualNet [105]. QualNet is a commercial product by Scalable Network Technologies and provides a complete suite for building complex network simulations including WLAN, WSNs, GSM/*Universal Mobile Telecommunications System* (UMTS), satellite links, Local Area Networks (LANs) and Wide Area Networks (WANs). It

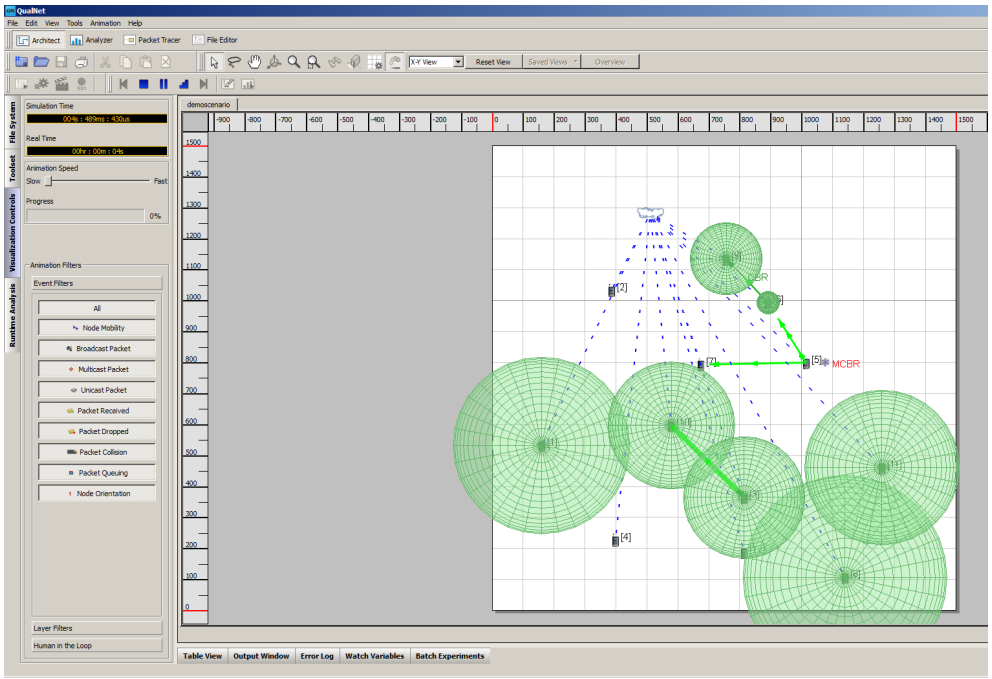


Figure 2.14: Screenshot of Qualnet executing a demonstration scenario.

features a complete replication of the ISO/OSI protocol stack[2] with all well-known protocols on all layers. QualNet is especially known for its precise implementations of the physical and medium access control layer of the protocol stack. The GUI allows to create simulation scenarios by dragging different components conveniently into the simulation area. When executing the simulation important events like broadcasts, unicast direction, packet drops, weather or node mobility are visualized to give a graphical feedback to the user. QualNet provides an API and interfaces for the user to integrate own protocols into the simulation. This is not restricted to the protocol stack itself, it is also possible to define new models for weather parameters or node mobility behavior. A screenshot of QualNet while executing a demonstration scenario is visualized in Figure 2.14.

Like other network simulation tools (e.g., ns2 [106], OMNeT++ [107]) QualNet separates the GUI from the simulation core. While the simulator itself is a command line application and only works on configuration files describing the network scenario, the GUI of QualNet is used to provide a convenient

[2]Precisely, it is an implementation of the 5-layer stack also known as TCP/IP Internet Protocol Stack

way to configure scenarios and visualize the simulation process and simulation results. The simulation core and the GUI run in separate processes and communicate via a data pipe. Unfortunately, the GUI itself is acting sluggish and running complex scenarios will waste much time on the visualization process. Therefore, it is recommended to execute the simulation directly on the command line.

As mentioned earlier QualNet separates functionality and models the network according to the ISO/OSI model. While a strict implementation of this model would only allow a layer to communicate with its predecessor and successor, in QualNet it is allowed to communicate between all layers. This allows developing and testing whole new protocol stacks and impressively demonstrates the power of QualNet.

QualNet is an event-based simulation tool. This means every action in the network simulation can be traced back to a single and unique event. For this reason, QualNet maintains a virtual clock and each event is assigned to exactly one discrete point in time. Typical examples of events are expiring timers, reception of a packet, node movement etc. QualNet maintains an event queue and processes all events in a first-in-first-out manner by redirecting them to their corresponding model. A model in QualNet is everything responsible for handling a certain type of event. For example, if in a WLAN scenario a packet needs to be passed from the physical to the medium access control layer of a node there will be an event generated and QualNet redirects this event to the 802.11[3] protocol model. Each model in QualNet has to provide functions for initialization, finalization and event handling. In the initialization phase the model allocates memory, assigns model parameters from the configuration files to variables and sets the model to operational mode. The event handling is responsible to handle all messages for the model. In a routing protocol model this would mainly affect handling route requests, handling data packet forwarding, etc. The finalization phase is used to free all allocated memory and write the model statistics to disk for later analysis. The QualNet GUI can then be used to analyze the statistics and compare calculated metrics between nodes.

[3]There is a different model for each of the 802.11 standards including 802.11a/b/g/p/ac

2.3.3 Network Testbeds

Although simulation is a big advantage when developing and testing new ideas, it cannot replace field tests on real hardware as the used models might not represent the entire environment in every detail. Therefore, after a successful evaluation in simulation has been conducted, it is desirable to perform further tests in physical testbeds. However, without industrial partners and authorization of property owners a large scale field test consisting of thousands of nodes is impossible. Therefore, in academic research test bed sizes of only 10 to 100 nodes are common [37].

The logistic expenses escalate if a mobile WSN field test has to be conducted. It is hard to think of simple solutions to provide mobility in a testbed. Although moving humans carrying a sensor mote are an option, it is very inconvenient for experimental subjects to constantly walk or even run in the testbed area to provide a constant node velocity. A solution to overcome this inconvenience is using radio controlled cars. Sensor motes attached to these RC cars certainly reach higher velocities than an average human could constantly provide and therefore can be used for mobile scenarios with lots of changes in the network topology.

Chapter 3

Related Work

This chapter reviews previous work on range-free localization and puts strong focus on the MCL algorithm, since this method provides the theoretical base of this thesis.

Contents

3.1 Common Range-free Localization Approaches

3.1.1 Centroid Localization

Centroid is a simple range-free localization scheme introduced by Bulusu et al [103]. Centroid uses a regular mesh of nodes which are aware of their positions and serve as reference points. These nodes send out location announcements called beacons in regular intervals. A simple connectivity metric based on packet counting is used to rate the connectivity between an ordinary node and its available reference points. In Equation 3.1.1 N_{recv} and N_{send} denote the amount of beacons received from and sent by the ith reference point in a period of t.

$$CM_i = \frac{N_{\text{recv}}(i,t)}{N_{send}(i,t)} \times 100 \tag{3.1.1}$$

Only reference points with a $CM > CM_{\text{thresh}}$ are considered for localization. The authors propose to set $C_{\text{thresh}} = 90$, i.e., only reference points with high delivery ratios are assumed to be connected to the node. The set of k chosen reference points is used to estimate the location of the node as given in Equation 3.1.2.

$$(X_{\text{est}}, Y_{\text{est}}) = \left(\frac{X_{i,1} + ... + X_{i,k}}{k}, \frac{Y_{i,1} + ... + Y_{i,k}}{k} \right) \tag{3.1.2}$$

The proposed centroid calculation approach is implemented on Radiometrix radio packet controllers, which allow sending arbitrary data over 418 MHz channels with a radio range of about 9 m. The approach is evaluated in a static outdoor scenario with 100 nodes organized in a grid structure of 10 m $\times$ 10 m with 4 reference points residing on the four corners of the grid. The setup is shown in Figure 3.1.

Figure 3.2 shows the absolute localization error of the Centroid evaluation. The figure illustrates a general problem of Centroid: Close to the corners where the reference point coverage is low due to the restricted connectivity the localization error is very high. Areas with good reference point coverage

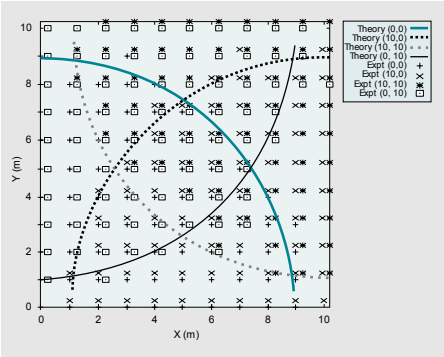


Figure 3.1: Centroid evaluation setup.

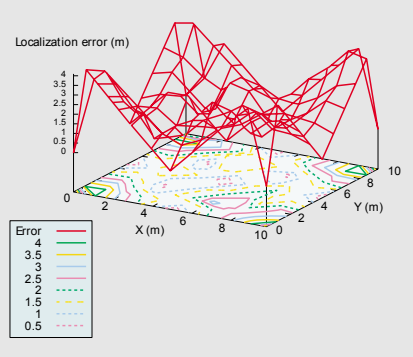


Figure 3.2: Centroid localization error.

achieve reasonable localization errors. The evaluation of Centroid is done only for static scenarios. Mobile sensor networks are not considered.

3.1.2 DV-Hop Localization and Amorphous

A technique called DV-Hop localization using trilateration without the need for active ranging is presented by Niculescu and Bath [108]. The key idea of DV-Hop is adapted from classic distance vector routing protocols. Location announcements originating from reference nodes (anchor nodes) are flooded through the network using broadcast transmission. In every packet a hop counter is maintained and increased at every hop. Every intermediate node stores the minimum value of the hop counter for each unique location announcement. Multiple received location announcements with equal or higher hop count are discarded. Using this mechanism eventually all nodes will have the shortest distance in terms of hop count to every reference point.

The task of converting the hop count to physical distance is also performed by the anchor nodes. Following Equation 3.1.3 the average hop distance of the ith anchor is calculated by dividing the sum of distances to all other anchors (1 to j) by the sum of hop counts.

$$\text{HopSize}_i = \frac{\sum \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}}{\sum h_j} \tag{3.1.3}$$

The calculated average hop distance HopSize_i for anchor i is then propagated in the network again. An ordinary node is able to localize itself using trilateration (see Section 2.2.4.1) upon it has received the average hop distance of 3 different anchor nodes.

DV-Hop can only give a very rough estimation of the position ϕ of a node, because the distances between nodes can highly vary and averaging the distances to other anchor nodes is not necessarily representative for the network topology.

Amorphous [109] is a localization algorithm which uses a similar hop counting technique like DV-Hop to provide every ordinary node with the number of hops to the anchor nodes in the network. However, the hop count from an ordinary node to an anchor node is calculated as the average of the neighboring hop counts, instead of relying only on the own calculated hop count.

The single hop distance is calculated offline using Equation 3.1.4. Amorphous assumes the average connectivity of the network n_{local} is known a priori and does not change during operation time.

$$\text{HopSize} = r\left(1 + e^{n_{\text{local}}} - \int_{-1}^{1} e^{-\frac{n_{\text{local}}}{\pi}\left(\arccos t - t\sqrt{1-t^2}\right)} dt\right) \tag{3.1.4}$$

Using the calculated hop distance an ordinary node can estimate the distance to all anchor nodes. Three or more anchor nodes are required to perform the least squares method to calculate the location estimation of the node.

Both DV-Hop and Amorphous have not been evaluated in mobile sensor networks. Especially Amorphous is not designed for changing network densities, as it requires to calculate the parameter n_{local} offline.

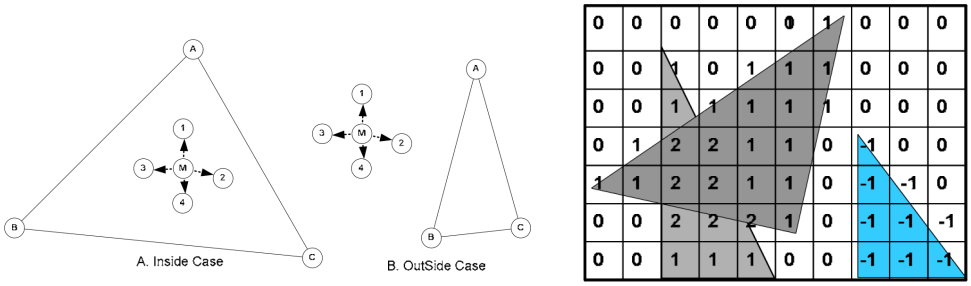
There exist numerous extensions of the presented algorithms above which often focus on more precise distance estimation between ordinary and anchor nodes. A common problem of all algorithms is their lack of mobility support and the requirement of at least three available anchor nodes.

3.1.3 APIT Localization

APIT [102] is a localization algorithm which uses combinations of received anchor information to form different triangles. For each triangle a node *M* performs a *Point-in-Triangle* (PIT) test to check if it resides inside or outside of this triangle. A geometrically exact test can only be performed if the position of *M* is known. However, this is not the case, since the location of *M* is unknown of course. The adapted *Adapted Point-in-Triangle* (APIT) test therefore uses neighboring information based on signal strength from both anchors forming the triangle and ordinary nodes to determine if there is a neighbor of *M* which is closer *to all three* anchors. If not, *M* is assumed to reside inside the triangle, otherwise it is considered to be outside of the triangle. The technique is illustrated in Figure 3.3(a). It is important to note that APIT uses the signal strength only to order distances relatively and no absolute distance estimations are performed. Otherwise it would be categorized as a range-based algorithm.

APIT exploits the high node density which can be expected in sensor networks. However, evaluation studies show that even with 6 neighbors there is still a false positive error of 16%, i.e., the node is assumed to be inside the triangle, although it is not. For lower node degrees no information about the error is given.

The APIT test is performed for all possible triangle combinations of the received anchor information. Only positive triangles, i.e., the triangles node M is assumed to reside in, are further considered. To narrow down the



(a) APIT test as shown in [102].

(b) APIT SCAN algorithm as shown in [102].

Figure 3.3: APIT techniques.

region where M resides, APIT uses a SCAN algorithm which divides the deployment area into a grid of cells with dimensions $0.1 \times r$ where r denotes the radio range. Initially, all cells are assigned with the value 0. The cells overlapping with triangles formed by anchors are incremented by 1 for each positive triangle and decremented by 1 for every negative triangle. The group of cells with the highest cell values is representing the intersection region of all positive triangles. The SCAN algorithm is illustrated in Figure 3.3(b). Following this, APIT calculates the *Center of Gravity* (COT) of the formed cell group, which is the final location estimation of node M.

APIT is evaluated via simulation and compared to the approaches introduced above. APIT outperforms Centroid, DV-Hop and Amorphous if the number of heard anchor nodes is at least 10. Furthermore, the evaluation shows a decreasing localization error for high connectivity values of at least 6 ordinary node neighbors.

3.2 Monte Carlo Localization

This section describes the algorithm this thesis is based on. Monte Carlo Localization is an approach which was originally proposed for the usage in robotics [110] in 1999. It was adapted for Wireless Sensor Networks by Hu and Evans in 2007 [36]. MCL introduced a novel concept for localization in WSNs and aimed to compensate for the comparatively high localization error of other range-free solutions while keeping reasonable algorithmic simplicity. Furthermore, MCL is designed to account for mobility of both seed and ordinary nodes. Other approaches like Centroid [103] or APIT [102] are designed for static networks which are not deployed in an ad hoc manner.

Monte Carlo methods make use of random sampling to approximate numerical results [111]. In MCL the position estimation of a node is represented by a set of weighted samples where each sample represents a possible location of the node. Impossible locations are filtered out using a particle filter which is updated based on seed node information. To account for growing uncertainty about the position of a node due to mobility, particles are moved arbitrary from their previous location. The mathematical background of this algorithm is to approximate the probability density function of the position

by a set of weighted particles (samples) and use importance sampling to eliminate less probable particles. To filter samples, a Bayesian filter is used which are often applied in situations in which new observations impact the state of the system.

The following sections explain MCL in detail and discuss existing approaches to improve the performance of MCL in both precision and computational overhead.

3.2.1 System Model and Known Parameters

In MCL all nodes are assumed to be mobile and able to move arbitrary in the deployment area. Seed nodes are equipped with GPS and can send location announcements to other nodes using broadcast messages. Nodes receiving these announcements handle them as new observations they made between two localization approaches. Furthermore, all nodes have a maximum velocity and a maximum communication range.

- $v_{\max}$
  The maximum velocity a node in the network can have
- r
  The communication range of all nodes in the network. Heterogeneity in the radio range is modeled using a variable which accounts for path fading effects, obstacles or other impacts which might affect the signal quality.

3.2.2 Design

In MCL a node's estimated location ϕ_{est} is represented by a set of weighted samples, L, where each sample l_t represents a possible location of the node at time t. The initial set, L_0, is selected by choosing random locations of the whole deployment area. A node will always maintain a fixed number of samples, N, to guarantee enough variability while still limiting the computational overhead. The MCL algorithm shown in Figure 3.4 computes the sample set L_t using the information of sample set L_{t-1} and observations

1: **procedure** MCL
2: $L_t = \{\}$
3: **while** $size(L_t) < N$ **do**
4: $R = \{l_t^i | l_t^i$ from $p(l_t|l_{t-1}^i)$, $l_{t-1}^i \in L_{t-1}\}$
5: $\forall\, i, 1 \leq i \leq N$
6:
7: $R_{\text{filtered}} = \{l_t^i | l_t^i$ where $l_t^i \in R \wedge p(o_t|l_t^i) > 0\}$
8: $L_t = \text{choose}(L_t \cup R_{\text{filtered}}, N)$
9: **end while**
10: **end procedure**

Figure 3.4: MCL algorithm in pseudo code.

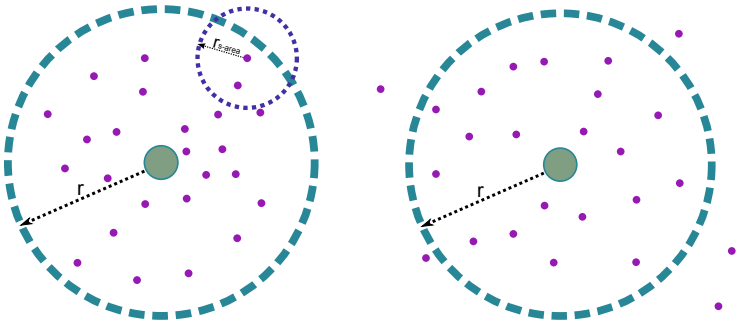
of seed nodes, o_t, available at time t. To account for uncertainty about the node movement behavior, the algorithm includes a *prediction step* (line 4) in which a new sample is drawn from a circular sampling area with radius $r_{\text{s-area}} = v_{\max} \times t_{\text{check}}$ around its current position given by a transition equation $p(l_t|l_{t-1})$. In MCL $v_{\max}$ is the maximum velocity of a node and t_{check} is the time between two localization attempts of a node. The probability of the current location based on the previous location estimation is given by a uniform distribution [36], where $d(..)$ denotes the Euclidean distance between two samples as shown in Equation (3.2.1).

$$p(l_t|l_{t-1}) = \begin{cases} \frac{1}{\pi \times {r_{\text{s-area}}}^2}, & \text{if } d(l_t, l_{t-1}) \leq r_{\text{s-area}} \\ 0, & \text{if } d(l_t, l_{t-1}) > r_{\text{s-area}} \end{cases} \tag{3.2.1}$$

The MCL prediction step is illustrated in Figure 3.5. Figure 3.5(a) illustrates a node with its current sample set and a single sample with its sampling area. For each sample a new sample is drawn in the circular area with radius $s_{\text{s-area}}$ will be drawn. The resulting sample set is shown in 3.5(b).

After the prediction step the generated set is put into the *filtering step* (line 7) which uses the observations o_t (i.e., location announcements from seed nodes) to filter impossible node locations from the sample set. Each node keeps track of its first-hop neighbor seeds S and of its second-hop neighbor seeds T. The filtering condition for a sample l is given in Equation (3.2.2).

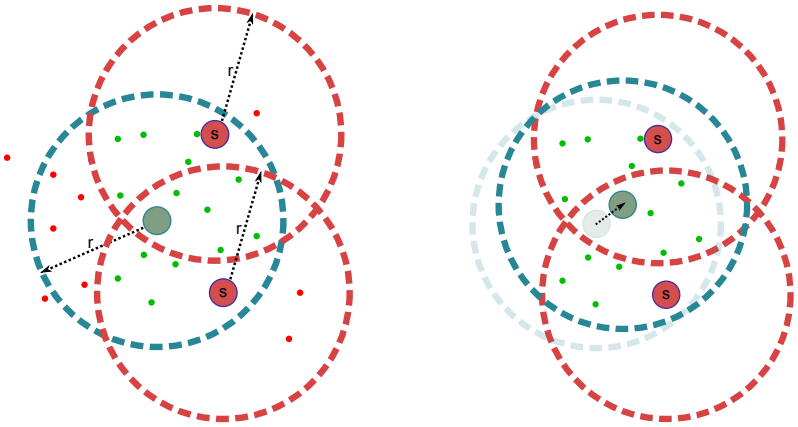
$$\text{filter}(l) = \forall s \in S, d(l,s) \leq r \wedge \forall s \in T, r < d(l,s) \leq 2r \tag{3.2.2}$$



(a) Sample set and sampling area

(b) Updated sample set after prediction step

Figure 3.5: MCL prediction step.



(a) Filtering process

(b) Location estimation update

Figure 3.6: MCL filtering step.

Samples passing the filter are assigned a weight of 1, all others a weight of 0. Samples with a weight of 0 are ignored in further re-sampling steps. If more than N samples have been generated, N random samples are chosen from the current set (line 8). The process is repeated until $|L_t| >= N$ (line 3). The final step is to compute the position estimation ϕ_{est} by calculating the weighted average of the sample set, i.e., the average of all samples with a weight of 1.

The process is illustrated slightly simplified[4] in Figure 3.6. In 3.6(a) the green samples are passing the filter condition, because they are located in the area

[4]Only first-hop neighbor seeds are considered

between the seed nodes and the ordinary node. The red samples will not be considered for the new location estimation which is calculated from the remaining particles as shown in 3.6(b).

3.3 Existing Improvements of MCL

Existing work on improvements of MCL mainly focuses on enhancing the filtering step either by introducing a more sophisticated sample weighting or by defining more precise filter conditions. In MCL the sampling area is defined by a circle around each sample and not affected by heard seed nodes. Baggio and Langendoen propose *Monte Carlo localization boxed* (MCB) [112] in which they are constructing an anchor box of all received seed nodes. New samples can only be drawn from this area. As a consequence, the probability of drawing good samples, i.e., samples that are matching the filter condition, is much higher compared to MCL. Although the evaluation shows that MCB is indeed benefiting from the anchor box, the approach assumes that at least two seed nodes are always present. Otherwise the anchor box cannot be constructed at all. Consequently, their evaluation does not present the behavior of MCB if the number of seed nodes is reduced.

Yi et al. propose *Multihop-Based MCL* (MMCL) [113], a combination of MCL, the DV-Hop localization method [108] and MCB [112] as described above. The advantage of MMCL is that location announcements are flooded through the whole network, so each node always has seed information available (as long as it is not completely isolated from the network). Each node keeps track of the hop distance to seed nodes and estimates the distance to anchors using formulas based on the hop count c_i and two parameters α and β which are very sensitive to changes and therefore mainly determine the quality of the distance estimation. Consequently, the parameter r of MCL is no longer required to be known and replaced by the new distance estimation method. Like in MCB, MMCL constructs a bounding box from where new samples are drawn.

Rudafshani and Datta [114] take neighboring information into consideration for calculating sample weights. This means not only seed nodes contribute to the localization process of a node, but all neighbors of a node do. The au-

thors present two versions of their algorithm: MSL* calculates weights for all of a neighbor's samples, while MSL tries to reduce the therefore emerging computational and communication overhead by calculating a single weight for every neighbor. Hence, in this work a closeness value is introduced which describes the quality of a location estimate. The closeness value is used to identify neighbors which seem to have a good estimation of their own position and therefore provide more valuable information to the node which tries to localize. The evaluation shows that using neighborhood information gives better results especially in situations where the seed degree is low.

Zhang et al. propose *Weighted MCL* (WMCL) [115], which provides further improvements for the MCL algorithm family. A bounding box is constructed to reduce the area from which new candidate samples are drawn. According to the authors this results in less overhead when the new sample set is computed. In addition to that, WMCL makes use of neighborhood position estimations to increase the localization efficiency. The authors state that their algorithm works better in static scenarios than previous solutions without tuning special parameters as done in MSL/MSL*. A real implementation on MICAz motes [69] is provided in a small testbed to evaluate a static scenario.

Teng et al. [116] show how a single mobile seed node can be used to localize all nodes in a static sensor network in their approach called *Mobility-Assisted Monte Carlo Localization* (MA-MCL). Applications for these scenarios might be rare and can only be expected if sensor nodes are deployed in a rush, i.e., there is no time to calibrate the position. The sensors are not expected to move after deployment. The seed node is the only node equipped with GPS and will move randomly in the whole deployment area. Since the unknown nodes do not move, they generate new samples based on their current position only in a static square area with side length β which is a tunable parameter of the algorithm and is standardly set to $\beta = 0.1r$ where r is the communication range of a simple node. The authors compare their work to the solution given in [114] and show that in these special scenarios their algorithm outperforms MSL and MSL*. However, they created a single point of failure situation: if the mobile seed fails for any reasons the whole network will not be able to localize. A similar work using a single mobile seed

node is presented by Huang and Záruba [117]. However, a fundamental difference is that the location of the nodes is calculated on the mobile node instead of the ordinary nodes. Furthermore, if available, ranging information like angles and distances measured using RSSI is included to achieve a better sample prediction. Putting the localization approach on the mobile nodes relaxes the computational overhead on the ordinary nodes, but creates an even more drastic single point of failure.

Chapter 4

RSSI as a Distance Estimator

In this chapter a study of a commonly used distance estimator is presented. Given a suitable path fading model the *Received Signal Strength Indicator* (RSSI) can be used to calculate the distance between a sending and a receiving node. Range-based localization approaches often rely on RSSI distance estimation, since this method does not require additional hardware and is an already present feature in wireless networks. However, the quality of the distance estimation primarily decides on the localization error. The presented study evaluates if relying on the RSSI as a distance estimator is a feasible method for the usage in localization techniques.

Contents

4.1 Motivation

The disadvantages of range-based localization algorithms as presented in Section 2.2.4 are mainly ascribed to costly or imprecise ranging methods. While this can easily be reasoned for AoA and ToA/TDoA methods, for ranging based on signal strength (RSSI) it is more difficult to point out problems.

The RSSI is an often used tool for estimating distances in WSNs, which is essential for range-based localization algorithms. Therefore, RSSI became one of the most commonly used distance estimators in scientific work [118, 119, 120]. While plentiful of algorithms make use of RSSI, only a few testbed studies on the quality of this estimator exist. Many of the conducted studies using network simulation or performed in controlled environments came to the conclusion that RSSI is a suitable distance estimator. However, studies in arbitrary environments and outdoor scenarios are rare. Multipath effects like scattering or reflection are often underestimated [121, 122]. To determine if RSSI might be a suitable distance estimator for the usage in localization algorithms, an extensive measurement study is conducted for this thesis. Before the experimental setup and measurement results are discussed, an overview of existing studies regarding RSSI distance estimation is given and several drawbacks and possible sources of impact, which might lead to varying measurements, are presented.

4.2 Previously Conducted Studies

While plentiful of localization algorithms exist which are based on distance measurement only a few studies on the quality of these estimations are available.

Adewumi et al. examine RSSI as a distance estimator in indoor and outdoor scenarios [33], but only with a maximum distance of 10 m. The authors do both simulations and a practical evaluation and emphasize differences between the optimal results of the simulation and their implementation on IRIS sensor motes [70]. They monitor RSSI readings over time, which shows

heavy variations. Unfortunately, the results are left uncommented. In the end, the authors use their collected data to calibrate the log normal shadowing model to have a perfect match for their measuring environment. Although their results look promising for short distances up to 5 m, the average estimation error is growing tremendously for larger distances.

A study of RSSI in an optimal indoor environment is provided by Parameswaran et al. [34]. They observe that RSSI measurement is related to the direction of measurement. For instance, measurements northwards will show different results compared to measurements eastwards. However, this might be related to antenna irregularities and is not further commented in the paper. Unfortunately, their study also only considers distances up to 10 ft. or approx. 3 m.

A detailed report on the existence of nonlinearities in well-known radio transceivers is given by Dieng et al. [35]. The paper derives the RSSI response curves experimentally by measuring in an indoor scenario with multiple devices and proposes a calibration method to get rid of the found nonlinearities. The authors conclude that application designers must keep in mind that RSSI response curves usually include nonlinearities and need to provide countermeasures.

Several works focus on studying the usage of RSSI for indoor localization. Chen and Terzis use a testbed composed of Tmote sky nodes to evaluate RSSI in a gird scenario with maximum distances up to 6 m [123]. They conclude that node orientation, i.e., antenna orientation, has heavy influence on the obtained results.

A similar study is provided by Benkic et al. with measurement distances up to 25 m [124]. Different transceiver modules are used and compared. The authors conclude that RSSI measurement is depending on the transceiver chip and the level of precision required by the application determines if RSSI can be used for distance estimation.

An opposite view, but with regard to link quality estimation, is provided by Srinivasan and Levis [125]. The authors revise RSSI measurements for link quality estimation and argue that newer chipsets as the CC2420 [74] provide a lower hardware calibration error. They examine the packet reception rate

and compare it to the average RSSI received and find a strong correlation up to the edge of sensitivity threshold. One could infer from these results that RSSI will also show a good performance when used for distances estimations.

A similar study is given by Wu et al. [126]. The characteristics of the RSSI in wireless local area networks are evaluated in different locations and show high variations. In a static indoor scenario RSSI is observed over time and shows high variations, too. The authors conclude that RSSI is subject to the common multipath effects (e.g., scattering, diffraction and reflection) and different locations will affect the RSSI quality drastically.

4.3 Limiting Factors of RSSI Quality

This section describes several sources of channel interference and limiting factors for the goodness of the RSSI readings.

4.3.1 Discrete RSSI values

Especially simulation studies of RSSI based distance estimation usually do not consider the fact that RSSI readings are discrete values in nowadays available transceiver modules. For upper layers of the protocol stack the RSSI is only available as a single register value with a value range of 1 Byte, i.e., 255 possible values at maximum. In practice the real value range depends on the manufacturer's specification and is often limited to a fraction of all possible values. All register readings must be converted to a value given in dBm following a formula given by the manufacturer. Often this is done by adding an offset value, which is different for every radio type. Table 4.1 shows an overview of some commonly used radio modules and their number of possible RSSI values. This means some radios can be used for more precise distance estimations compared to others. For instance, the SX1211 can be used to produce ≈ 4.5 times more accurate results compared to the AT86RF230. High resolution chips provide values with a granularity of up to 0.5 dBm, a more common value is 3 dBm. While value range and granularity can be reasonable looking numbers in data sheets, the most important variable for the quality of distance estimation is accuracy. Not all

Radio Type	Number of possible values
Texas Instruments CC2420 [74] CC2430 [127] CC2520 [128]	≈ 100
ATMEL AT86RF230 [129]	27
Infineon TDA5250 [130]	≈ 64
Telegesis ETRX2 [131]	40
Semtech SX1211 [132]	≈ 140

Table 4.1: RSSI value ranges of commonly used radios.

vendors publish information about the accuracy of the RSSI value readings. A common value is +-6 dBm, which means that RSSI values measured at a fixed distance fluctuate up to 12 dBm.

4.3.2 Path Loss Models

An important factor for the quality of the distance estimation using RSSI measurements is the path loss model, which puts the measured power levels in relation to the distance between sender and receiver and possible other factors like antenna gain or environment parameters. Often, inappropriate path loss models are used, which have not been designed for the usage in WSNs or require additional factors, which need to be found empirically for every application scenario. For instance, the log normal shadowing model [91] as shown in Equation (4.3.1) requires a propagation exponent n, which is depending on the environment and the reference RSSI between sender and receiver at distance 1 m. The chosen parameters are only valid at the time of calibration and might lead to decreasing accuracy at a later point in time if the environment conditions change. Furthermore, log normal shadowing and Friis free space propagation model [92] as shown in Equation (4.3.2) have both been designed for large distances of hundreds of meters as occurring in mobile telecommunications and therefore do not apply to the short range communication used in WSNs. The problem has been recognized by researchers and more suitable path loss models have been developed [93, 94]. However, until now, they are rarely used in scientific work.

$$P_r = A - 10 \times n \log d \tag{4.3.1}$$

$$P_r = \frac{P_t}{4\pi d^2} G_t \times \frac{\lambda^2}{4\pi} G_r \tag{4.3.2}$$

4.3.3 Weather Effects

Another effect, which is usually not considered in RSSI evaluation studies, is weather influence [133, 134, 135, 136]. The most important aspect of weather for radio propagation is moisture, i.e., phenomena like fog, rain and snow fall. While rain and snow add scattering and reflection depending on the intensity, air moisture itself leads to additional signal attenuation. Existing evaluations usually lack testing under different air moisture conditions and scattering influence added by heavy snow or rain is not considered at all.

4.3.4 Hardware Limitations

Physical limitations of the hardware components have strong impact on signal propagation. For instance, in simulation studies researchers tend to assume antennas allow perfect isotropic signal emission and radio transceiver modules provide exactly linear RSSI readings, which both holds not true in real-world scenarios [137]. In WSNs motes are often designed for one-way applications. Therefore, cost restrictions for cheap and lightweight sensor nodes do not allow the usage of high quality components. In mobile scenarios RSSI measurements are also affected depending on the velocity of the sensor mote.

4.3.5 Obstacles

Almost every path loss model assumes clear *line of sight* (los) communication between sender and receiver. However, in practice this is almost never the case. Wireless communication is always subject to signal propagation effects as reflection, scattering and diffraction. In addition, obstacles like trees, rocks or cars add attenuation to the signal, which results in drastic changes

of measured power levels at the receiving end [138]. As a consequence, the RSSI measured at the receiver is heavily distorted and does not represent the distance between the communicating nodes any longer.

4.4 Implementation and Evaluation System Setup

The evaluation is based on TinyOS 2.x [139, 140] running on CM3000 [71] sensor motes built by advanticsys [141]. As described in Section 2.1.5.4, these motes are equipped with TI CC2420 [74] radio transmitters and external 5 dBi antennas to enable long range distance measurements. Initial tests with these antennas resulted in transmission ranges of up to 350 m given clear los. The CM3000 sensor mote is chosen, because it is equipped with the most commonly used radio, which provides a large resolution of the RSSI register as shown in Table 4.1. Since the value range of the RSSI depends on the radio transceiver chipset as described in section 4.3, it is desirable to use sensor motes which are capable of providing a sufficient resolution.

4.4.1 RSSI Measurement Implementation

TinyOS applications are programmed using *network embedded systems C* (nesC) [140], a programming language similar to C, but with special properties matching the requirements of limited platforms like sensor motes. nesC is event-driven, i.e., the program is only reacting to incoming events and remains in sleep mode otherwise. Typical events are the reception of a packet, a triggered timer or new sensor data ready ready to be processed.

Listing 4.1 shows the code for initiating a new measurement series. In Measure.start(void) a timer is initialized to regularly trigger a new event which is caught by the procedure Timer.fired(void). A random delay is used to avoid congestion if multiple sources are initialized at the same time. Timer.fired(void) will set up dummy data packets only containing the measurement series ID and packet number as the payload. The packets are sent to the receiving node which will measure the RSSI and forward the results to the base station. The timer is stopped, if the maximum number of packets to send is reached.

```
command error_t Measure.start(void) {
    uint32_t o;
    if (!radio_started || running) {
        return FAIL;
    }
    running = TRUE;
    o = call Random.rand16();
    call Timer.startPeriodicAt(call Timer.getNow() + o, config.
        interval);
    return SUCCESS;
}

event void Timer.fired(void) {
    nx_struct measure_msg *msg;
    if (radio_busy) {
        return;
    }
    msg = call Send.getPayload(&pkt, sizeof *msg);
    msg->measure = config.measure;
    msg->counter = counter;

    /* send an empty packet */
    if (call Send.send(config.partner, &pkt, sizeof *msg) ==
        SUCCESS) {
        radio_busy = TRUE;
    }
    /* stop measuring if limit reached */
    if (config.count && ++counter >= config.count) {
        post stop();
    }
}
```

Listing 4.1: Measurement series control code.

Listing 4.2 shows how the final RSSI value of a packet is retrieved at the receiving node. Depending on the chipset of the transceiver a certain *RSSI_OFFSET* has to be added to bias the retrieved value. In case of the TI CC2420 the value for *RSSI_OFFSET* is -45 [74].

```
event message_t *Receive.receive(message_t *msg, void *payload,
    uint8_t len) {
    int8_t rssi;

    nx_struct measure_msg *p = payload;

    /* get sender node ID */
    am_addr_t source = call AMPacket.source(msg);

    /* is the message OK (a packet from our communication partner
       )? */
    if (len == sizeof *p && source == config.partner) {
        rssi = call RssiPacket.getRssi(msg) + RSSI_OFFSET;
        signal Measure.received(p->measure, p->counter, rssi);
    }

    return msg;
}
```

Listing 4.2: RSSI extraction from received packet.

4.4.2 General Measurement Setup

For all measurements the same two communicating motes are used to avoid measurement variations due to different hardware characteristics. All results are sent to a central base station, which is connected to a laptop to record the output and also operates as the control station to start and stop new measurement series. Figure 4.1 illustrates the general setup. Initial tests resulted in a maximum communication range of about 350 m in outdoor scenarios with clear line of sight. To keep a sufficient level of packet reception, only measurements up to 300 m are conducted. In all experiments the motes are static, i.e., no experiments under the effect of mobility are studied. During measurement the motes are placed approximately 1 m above ground. A student moves the mote to the next measurement point. At every measurement point both motes exchange 1000 packets to calculate the average RSSI. The same two communicating motes with the same antennas are used in every experiment to avoid variations envoked by different physical properties. Consequently, if not stated otherwise, RSSI measurements are only evaluated unidirectional.

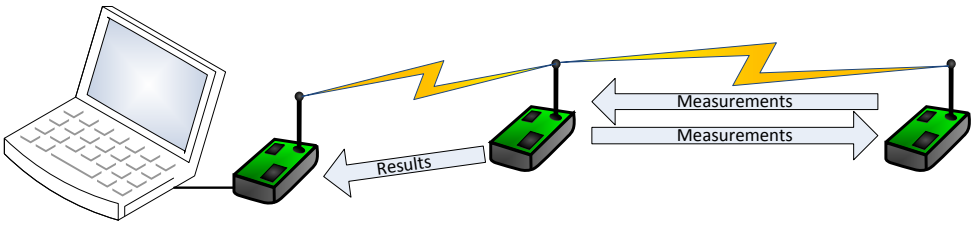


Figure 4.1: General RSSI measurement setup.

4.4.3 Evaluation Scenario Test Setup

The least interference-prone scenario to achieve the most reliable results is most likely the clear line of sight scenario, as signal attenuation introduced by obstacles and multipath effects is reduced to a minimum. The experiment is illustrated in Figure 4.2. The test track of 300 m is divided into intervals of 25 m. Clear line of sight is guaranteed during the experiment for the whole distance. For each measurement one of the motes is moved to the next measurement point, while the other one remains static at the beginning of the test track. The experiment is repeated on different days and different times to examine the effect of changed weather conditions.

A different test setup is used to analyze the effect of obstacles. Figure 4.3 illustrates how a building is used to introduce heavy signal attenuation. Due to the local conditions, in this experiment a maximum test track distance of only 75 m is available. It is obvious that the antenna signal has to run through the building or arrive via multipath effects at the receiving node. Indoor scenarios are tested in two long hallways with clear line of sight. These hallways differ in height and width and are filled with different furnishings. Again, a test track with a maximum distance of 75 m between two sensor motes is available.

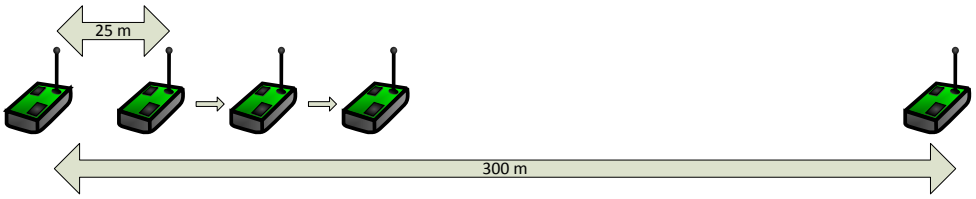


Figure 4.2: Experimental setup for measurements with clear line of sight.

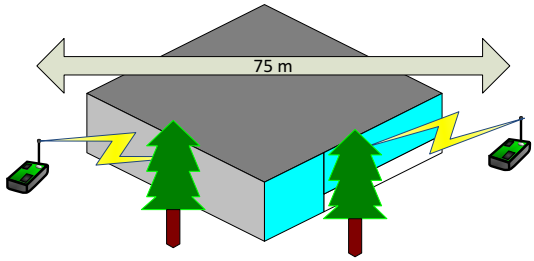


Figure 4.3: Experimental setup for measurements with obstacles.

4.5 Evaluation Results

4.5.1 Long Distance Measurement with clear los

Figure 4.4 shows the RSSI values measured in four different series. These measurements are done on multiple days under different weather conditions including snow and light rain. It is obvious that there is a general trend for all measurement series showing a loss of signal strength for larger distances. However, the curves are not monotonically decreasing, although they represent the average of 1000 collected RSSI values for each measurement point. This can be explained due to multipath effects: some measurement spots are benefiting from more advantageous reflections even if they are further afar.

The second observation is that values at the same measurement point, but for different measurement series, differ up to 10 dBm. In the context of distance estimation for range-based localization this means that at different

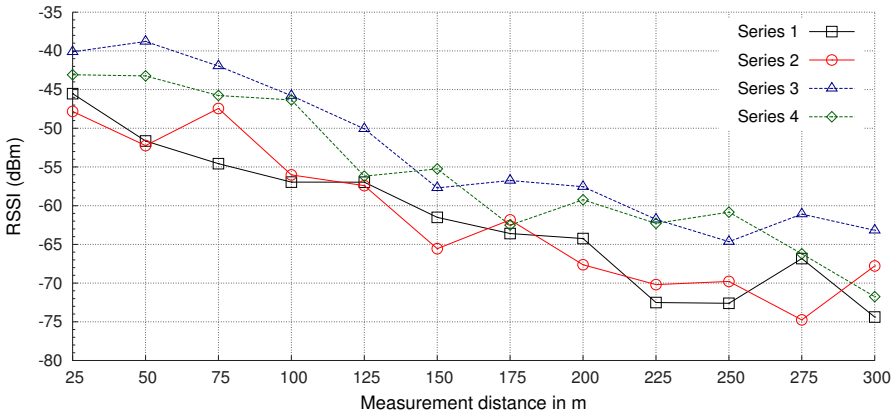


Figure 4.4: RSSI readings for long distance clear line of sight measurement.

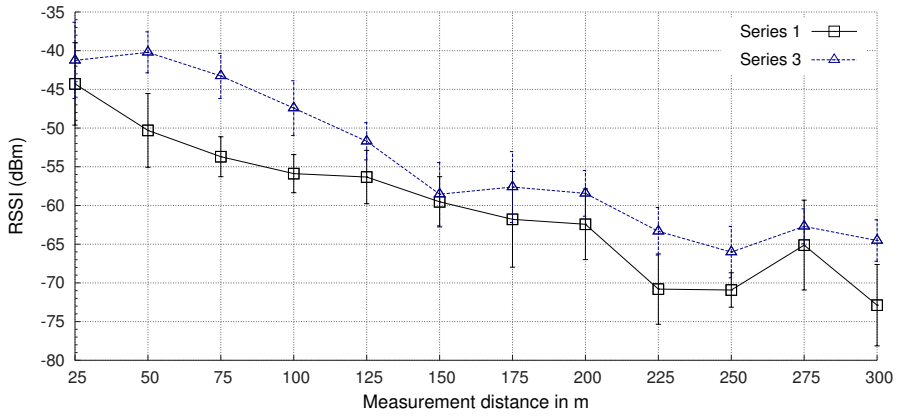


Figure 4.5: RSSI readings with mean deviation.

points of time the distance estimation can also vary by the factor 10. Most likely this can be explained due to different weather situations and changed objects like differently parked cars in the environment which results in distinct multipath effects for each measurement series. In summary, no reliable information suitable for distance estimation can be obtained using RSSI measurements in this scenario.

4.5.2 Standard Deviation in Long Distance Measurement

Figure 4.5 shows measurement series 1 and 3 plotted with error bars illustrating the standard deviation for each measurement point. Even for 1000 collected samples the measurements vary up to 6 dBm. The measurements confirm the information given in the product manual of the CC2420 [74], which states that RSSI readings may vary by about ± 6 dBm. This fact carries a great weight for the usage of RSSI in distance estimation, as it indicates that single measurements can barely provide reliable information. However, almost all localization algorithms based on active ranging rely on beacon nodes, which send single beacon packets in fixed intervals or retrieve the RSSI from data packets of the common network traffic.

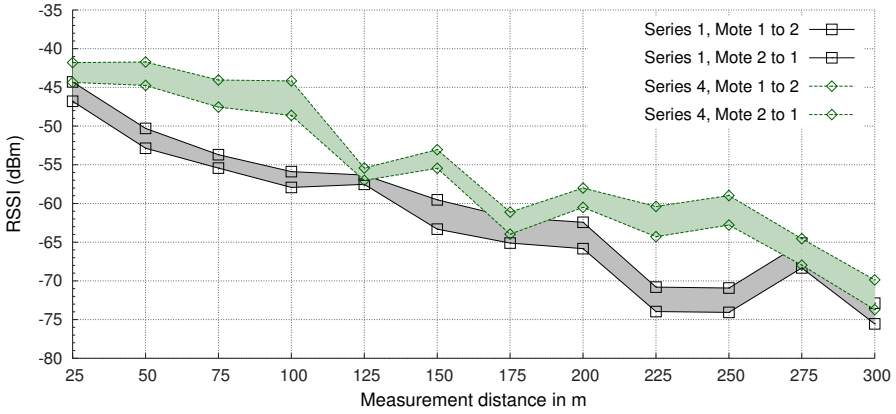


Figure 4.6: RSSI readings affected by physical hardware properties.

4.5.3 Effect of Hardware Characteristics

As stated earlier, hardware differences introduced in the manufacturing process or during deployment of the sensor motes can cause variations in RSSI readings. Figure 4.6 illustrates this by visualizing the graphs for both communication directions for measurement series 1 and 4. The hardware components used in this evaluation are officially of identical construction, i.e., the same antennas, chipsets and minor electronic components were assembled to built the sensor mote.

In Figure 4.6 it is easy to observe that the signal strength is following the same profile in both directions, however, there is always a discrepancy of about 4 dBm. The experiment is repeated with other hardware to verify this behavior. All tested motes show similar curves, but the difference in RSSI readings varies from 1 dBm to 4 dBm. Consequently, to rely on RSSI readings, one has to ensure that all used hardware components have the same characteristics and give the same RSSI readings for equal distances, i.e., calibration is required.

4.5.4 Effect of Obstacles

Figure 4.7 illustrates the signal behavior of two measurement series under the effect of signal attenuation due to buildings or natural barriers like rocks or trees. The averaged results of the previously presented clear los mea-

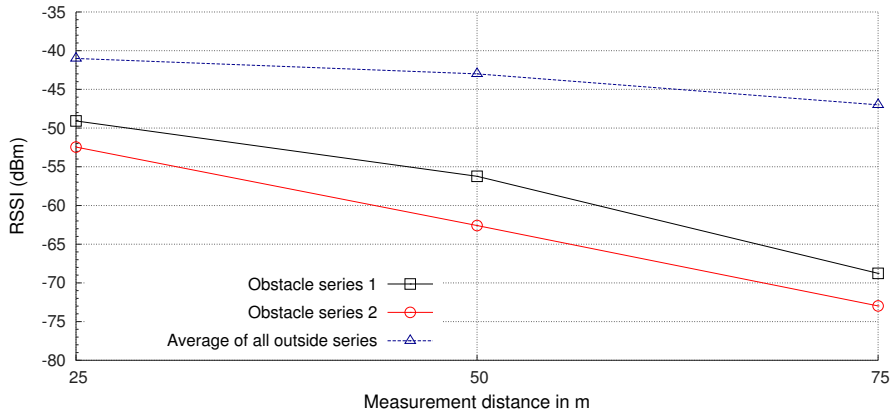


Figure 4.7: RSSI readings for measurements with obstacles.

surement series is given as a reference. Obviously, there is a big difference between measurements with and without obstacles. The signal is attenuated by up to 22 dBm, which means the power levels of the incoming signal are 100 times lower compared to measurements without obstacles. Therefore, distance estimations based on these readings would be subject to a very high error. In addition to that, the effect of changed multipath characteristics can be observed in this experiment, too. The readings of series 1 differ up to 10 dBm from series 2, i.e., no steady information can be obtained.

4.5.5 Effect of Indoor Scenarios

An often mentioned advantage of GPS-free range-based localization solutions is their ability to work in indoor scenarios, too. However, most of the time the proposed solutions do not account for the changed signal propagation characteristics of indoor environments. Figure 4.8 illustrates two indoor measurement series compared to the average of all outdoor series. Obviously, in indoor environments the signal strength is clearly lower, although clear los is provided at all times. In addition to that, different measurement series also differ depending on the multipath characteristics of the environment. Different furnishings and building materials can lead to big gaps between RSSI readings. Again, the RSSI does not provide a reliable source of information for distance estimations in WSNs.

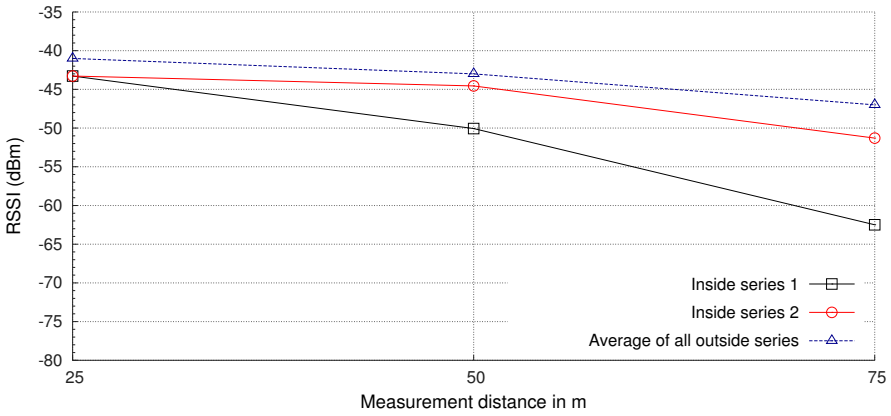


Figure 4.8: RSSI readings for inside measurements.

4.6 Summary

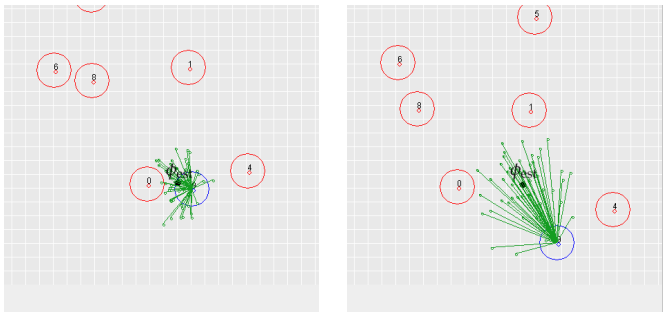
In this initial study of the thesis the quality of RSSI as a distance estimator was examined. Several measurement scenarios were designed and evaluated based on Advanticsys CM3000 motes with the TI CC2420 transceiver chipset. The outcomes of the evaluation confirm the initial concerns and show the insufficient performance of RSSI as a distance estimator for range-based localization. Although a general trend showing the coherence between distance and RSSI can be noticed, the readings are too unsteady to rely on them for distance estimations with a reasonable low error. Since localization algorithms may not introduce significant communication overhead in the network, distance estimation based on RSSI is usually done using the ordinary network traffic or only for location announcement packets. However, the results show that even with 1000 averaged RSSI packets no reliable estimation can be generated. As a consequence, further conducted research in this thesis concentrates on range-free localization solutions.

Chapter 5

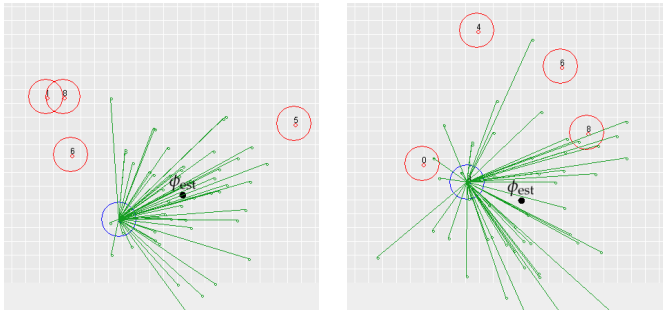
Sensor-Assisted MCL

This chapter presents *Sensor-Assisted Monte Carlo Localization* (SA-MCL), a new variant of the MCL algorithm, which accounts for situations where no location announcements have been received by a node. In SA-MCL additional sensor information is used to bypass the problem of missing seed information occurring in these situations, while keeping the localization error low. This is achieved by determining heading and velocity of a node and updating its position based on this information. The approach is evaluated in extensive simulations and in a real-world field test. The results show that SA-MCL can successfully reduce the localization error by up to 60%.

Contents



(a) t=5 (b) t=10



(c) t=15 (d) t=20

Figure 5.1: Sample set degeneration of MCL.

5.1 Motivation

The Monte Carlo Localization algorithm as introduced in Section 3.2 is a robust and easy to implement localization solution. However, as any other algorithm based on anchor information, the precision of the algorithm mainly depends on the number of available seed nodes. Mobility in the network can generate situations and topologies where single simple nodes are isolated or only have intermittent contact to seed nodes. In this case, the sample set of MCL will degenerate due to MCL's prediction step, which will spread the samples gradually over the whole deployment area. As a consequence, the location estimate, ϕ_{est}, is getting increasingly imprecise the longer a simple node has no contact to seed nodes. Figure 5.1 shows screenshots of the GUI representation, which has been developed for the Java simulator used to evaluate MCL by Hu and Evans [36]. The green lines indicate the distance from the node to its spread samples. While shortly after losing contact to

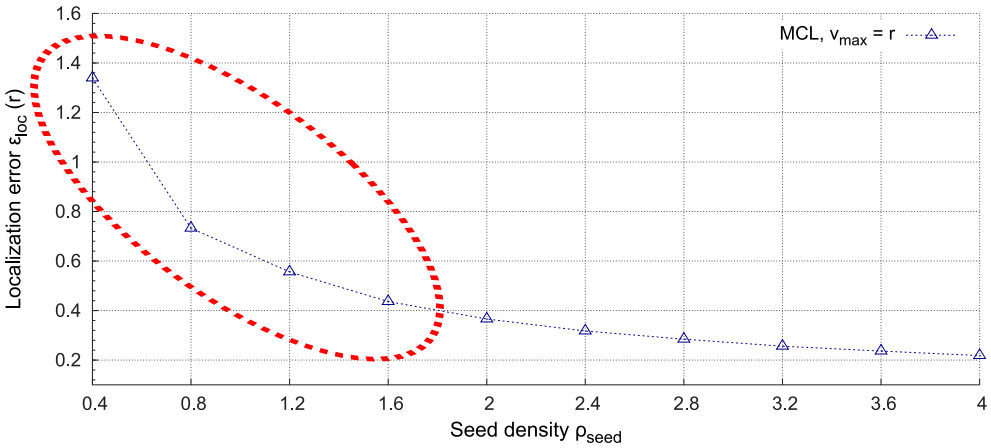


Figure 5.2: Localization error for different seed node densities.

all seed nodes (red circles) the sample set of the node (blue circle) is still compact, it quickly degenerates which results in a worse ϕ_{est} as indicated by the black dot. Eventually, in longer periods of seed information absence the samples will spread over the whole deployment area. Consequently, a node will be localized in the center of the deployment area in this case.

Figure 5.2 illustrates the problem from another point of view. Here, the localization error of MCL is plotted for different seed densities, i.e., the number of seed nodes available for localization for each node. It is obvious that for lower seed densities the localization error is much higher, as situations without seed information occur more often. The localization error is subject to exponential growth if the seed density is further decreased. Nevertheless, network operators have a strong interest in using as few as possible seed nodes to save deployment costs and to reduce the maintenance overhead of the network. SA-MCL addresses both problems by implementing a dead reckoning approach in situations where seed information is not available.

5.2 Design

To overcome the issues mentioned above, in *Sensor-Assisted Monte Carlo Localization* (SA-MCL) additional sensor information is used to update the MCL sample set, in particular, by adding the distance traveled by a node from the last location estimation to all samples, instead of executing

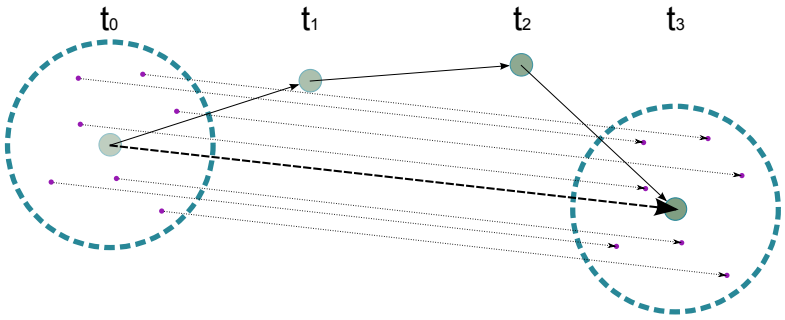


Figure 5.3: Relative particle movement to last estimated position.

MCL's prediction step. In this case, the state of the sample set is frozen, i.e., the samples do not spread, until the node is receiving new location announcements (i.e. moving in the communication range of seed nodes) again. Consequently, ϕ_{est} will be further updated even if no new observations are available. The additional data required to determine the distance and direction from the last known location is the node's velocity and its heading, i.e., its current orientation, respectively. Both values can be determined using common ots sensors:

- A **magnetometer** is a simple compass module able to measure the strength of the magnetic field of the Earth. Consequently, it can be used to determine the heading of an object in degrees where magnetic North is fixed at 0°.
- An **accelerometer** is a device able to measure proper acceleration, i.e., the acceleration relative to free fall. Accelerometers usually provide 3-axis-measurement and therefore are suitable for usage in three-dimensional space navigation. By integrating the accelerometer measurements the velocity of a moving object can be determined [142, 143].

The key idea of SA-MCL is illustrated in Figure 5.3. In this example the last position estimation occurred at time t_0. After that, the node loses contact to all seeds. The node is moving further and records sensor information at t_1, t_2 and t_3. At t_3 it is also supposed to determine its location again. From its recorded sensor information the node can calculate the distance and direction of movement relative from the last location estimation. Therefore, it is able to move the whole sample set according to the calculated velocity and heading. The technique of calculating a new location relative to a given one is called dead reckoning.

1: **procedure** SA-MCL
2: $L_t \leftarrow \{\}$
3: **if** $|o_t| < 1 \wedge sensorsActive$ **then**
4: $\Delta x \leftarrow$ getMovementXfromSensors()
5: $\Delta y \leftarrow$ getMovementYfromSensors()
6: $\forall\ l_t \in L_t$ do
7: $l_t.x \leftarrow l_t.x + \Delta x$
8: $l_t.y \leftarrow l_t.y + \Delta y$
9: **else**
10: **while** $size(L_t) < N$ **do**
11: $R = \{l_t^i | l_t^i$ from $p(l_t | l_{t-1}^i), l_{t-1}^i \in L_{t-1}\}$
12: $\forall\ i, where 1 \leq i \leq N$
13:
14: $R_{\text{filtered}} = \{l_t^i | l_t^i$ where $l_t^i \in R \wedge p(o_t | l_t^i) > 0\}$
15: $L_t = choose(L_t \cup R_{\text{filtered}}, N)$
16: **end while**
17: **if** $|o_t| < 2$ **then**
18: $sensorsActive \leftarrow true$
19: **else**
20: $sensorsActive \leftarrow false$
21: **end if**
22: **end if**
23: **end procedure**

Figure 5.4: SA-MCL algorithm in pseudo code.

In theory, dead reckoning allows to keep track of the own location given an initial position. However, over time sensor errors accumulate, which results in increasingly imprecise location estimations. Furthermore, it is desirable to turn off the additional sensor used in SA-MCL to save energy. SA-MCL is assuming that a node can easily keep track of direction changes and its velocity and therefore determine the traveled distance between two location estimates. A variety of practice-approved sensors exist, which provide the required data for SA-MCL [144, 145, 146].

5.2.1 Formal Description

The pseudo code of SA-MCL is shown in Figure 5.4. The main difference to the pseudo code of MCL as presented in Chapter 3 is that SA-MCL keeps track of the number of observations, $|o_t|$, i.e., the number of location announcements the node received. If no new observations are obtained, the

sensor information will be used to calculate Δx and Δy. MCL's prediction step to retrieve the intermediate sample set, L_t, is replaced by the SA-MCL method. Therefore, to obtain L_t these values are added to all samples in the old sample set, L_{t-1}. Otherwise, the default MCL algorithm is executed. Since using additional sensors cost energy, magnetometer and accelerometer are only activated if necessary. This is the case if the number of observations tends to reach 0. Once the node receives new location announcements, the sensors can be turned off again.

5.3 Implementation

SA-MCL is implemented in the simulation environment provided in [36] as described in detail in Section 2.3.2. The original Java code has been provided by Hu and Evans and can be retrieved from the website [147] of the University of Virginia.

Figure 5.5 shows how SA-MCL can be wrapped around an existing MCL implementation easily. As described above, only in the absence of seed nodes SA-MCL is executed. Activation and deactivation of the sensors is not implemented, since the network simulator does neither provide any emulation of real sensor mote components nor does it feature any energy consumption analysis. The implementation simply assumes the additional sensor data is present.

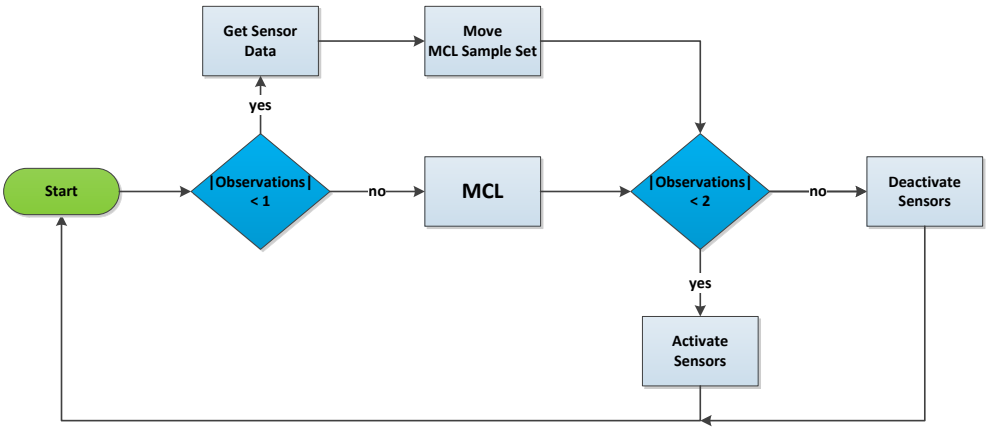


Figure 5.5: Flow diagram of SA-MCL

5.3.1 Scenario Execution

The simulation engine is step-based, i.e., the simulation process is divided in discrete steps which are separately executed. As shown in Listing 5.1 varying parameters can be simulated in a batch-like mode by looping over them. In the given example, the maximum velocity of the nodes *maxv* is evaluated. The outer for-loop increases the parameter *maxv* by 10 units per run. The number of iterations in the next for-loop determines how often a single experiment is repeated. The number of steps determines how long a single execution of a scenario lasts. When executing the scenario the first task is to update the locations of all nodes. After that, the code for MCL/SA-MCL is executed separately for every simple node and the current position error is calculated. All nodes are organized in a single array, the variable *start* determines the index of the first simple node. Since MCL and SA-MCL share the same node objects they also share the same sample set. Therefore they have to be executed separately since both algorithms would affect the sample set if run concurrently. Maintaining separated sample sets for each algorithm is a possible solution to avoid this problem. However, to avoid additional code modifications the sample set is shared and both algorithms are executed isolated from each other.

```
public void doAlgorithm() {
 double[] errorSum = new double[51];
 /* loop through all parameter settings */
 for(int l=0; l<radiorange.length; l++) {
  for(nodenum = 200; nodenum <= maxNodeNum; nodenum += 100) {
   for (maxv = 20; maxv <= maxMaxV; maxv += 10) {
    for (seednum = 100; seednum >= minSeedNum; seednum -= 10) {
     final_MCL = 0;
     final_SAMCL = 0;
...
     for (int curIt = 0; curIt<iteration_num; curIt++) {
      in.setParameter("max_v", Integer.toString(maxv));
      in.setParameter("node_num", Integer.toString(nodenum));
      in.setParameter("seed_num", Integer.toString(seednum));
      in.setParameter("node_r", Integer.toString(radiorange[l]));
      in.setParameter("node_s", Integer.toString(radiorange[l]));
      //only used in SA-MCL
      in.setParameter("max_sensorError", Double.toString(
```

```
            maxSensorError));
        //only used in PO-MCL
        in.setParameter("mag_freq", Integer.toString(
            magnetometerFreq));
        net = new Network(in);
...
        start = net.seed_num;
        end = net.node_num;
        for (int step = 0; step < step_num; step++) {
         net.updateLocation();
         MCL_error = 0;
         SAMCL_error = 0;
...
         for (int i = start; i < end; i++) {
          /* execute algorithms as desired */
          net.node[i].MCLocalization(net.relations[i], net.
              seed_positions, net.group_ref);
          MCL_error += net.statistics(1, i, step) / (end - start);
          //net.node[i].SAMCLocalization(net.relations[i], net.
              seed_positions, net.group_ref);
          //SAMCL_error += net.statistics(1, i, step) / (end -
              start);
         }
...
         for (int i = 0; i < Network.node_num; i++) {
          net.node[i].random_waypoint();
        }

        if (step >= stable_step) {
         net.avg_MCL_error += MCL_error / (step_num - stable_step);
         net.avg_SAMCL_error += SAMCL_error / (step_num -
             stable_step);
        }
       }
...
        final_MCL += net.avg_MCL_error / iteration_num;
        final_SAMCL += net.avg_SAMCL_error / iteration_num;
...
```

Listing 5.1: Scenario execution code.

5.3.2 SA-MCL Modification

In a simulation environment the exact positions of all nodes are known, since the implementation of the chosen mobility model is responsible for maintaining the movement behavior of the nodes, which includes waypoints and positions at any time during simulation execution. Therefore, it is very easy to implement the ideas of SA-MCL. Since it is naive to assume that heading and velocity of a node can be determined exactly, a sensor error is introduced in the implementation. This error is given by a random variable and accounts for imprecise sensor readings. Listing 5.2 shows the code section of SA-MCL where the sensor error is applied to the otherwise exactly determined distance to the last position estimation. SA-MCL is only executed if both iterators (*e* and *e2*) do not contain any elements, i.e., the node has not received any location announcements from seed nodes during the last interval.

```
if (!e.hasMoreElements() && !e2.hasMoreElements()) {
 int xNow = this.real_position.x;
 int yNow = this.real_position.y;
 int xDiff = xNow - this.last_time.real_p.x;
 int yDiff = yNow - this.last_time.real_p.y;

 //calculate error in range of -maxSensorError to maxSensorError
 double error =
  (Math.random() * 2 * this.maxSensorError)-this.maxSensorError;

 Random r = new Random();
 int sign = (int)(r.nextBoolean() ? 1 : -1);

 for (int m = 0; m < last_time.sample_num; m++) {
  Point p = last_time.sample_points[m];
  // calculate new sample position with respect to maxSensorError
  p.x += xDiff + xDiff*this.maxSensorError*sign;
  p.y += yDiff + yDiff*this.maxSensorError*sign;
  this_time.sample_points[m] = p;
  this_time.sample_weight[m] = last_time.sample_weight[m];
  this_time.sample_num++;
 }
}
```

Listing 5.2: SA-MCL Java implementation.

5.4 Simulation Evaluation Setup and Results

5.4.1 Simulation Parameters and Scenario Setup

The advantage of evaluating SA-MCL in the same simulator as MCL is that the results can be directly compared without the need to port MCL to another environment. On the other hand, the custom Java simulator lacks important features of wireless network simulation like path fading effects and only provides a rudimentary implementation of the random waypoint model. This both leads to more unrealistic simulations, since the real-world approximation is getting more abstract. Another issue of this simulator is its usage of abstract discrete units for time and distance with no association to commonly used physical units. As a consequence, time is expressed in simulation steps and estimation errors are given in multiples of the communication range of a node. The node velocity is also given as a multiple of the radio range r. A maximum velocity, $v_{\max}$, of 0.4 means between two localization attempts the node will move $0.4 \times r$ units in the simulator. Despite all disadvantages, the ability to directly compare SA-MCL to the original implementation of MCL prevails. For the ease of presentation the unit of distance metering is fixed to be meters in this thesis.

Sticking close to the evaluation of MCL as described in [36] the evaluation is performed with 300 nodes trying to localize themselves on a simulation area of $A_{sim} = 500\,\text{m} \times 500\,\text{m}$. The communication range for all nodes including seed nodes is fixed at 50 m. To simulate mobility in the network, a slightly modified version of the random waypoint model is used as explained in [36]. The only difference to the original model as described in Section 2.1.8.1 is that the pause time of all nodes is set to 0, i.e., when arriving at a waypoint the node will directly continue to move on to the next one. The authors of [36] changed this parameter to avoid longer periods of stationary nodes in which MCL would perform poorly, because of missing seed information. All experiments are repeated 50 times to truncate statistical outliers. The results presented below are the average of all experiments.

Table 5.1 lists the different simulation parameters which have been examined in order to evaluate SA-MCL. In this evaluation the total number of nodes,

Simulation parameter	Meaning	Default value
N_{nodes}	Total number of nodes	300
ρ_{seed}	Density of seed nodes	1.6
v_{max}	Maximum velocity of nodes	0.4
N_{sample}	Number of maintained samples	25
r	Radio range	50
ϵ_{sensor}	Error in sensor readings	20 %

Table 5.1: Evaluated simulation parameters for SA-MCL.

N_{nodes}, is 300, which includes all seed nodes. Obviously, the amount of seed nodes available during the localization process has high impact on the quality of the position estimation. SA-MCL is designed to handle scenarios with a reduced number of seed nodes, as explained in Section 5.2. Therefore, one of the most important parameters to study is the number of seed nodes available in the network.

It is difficult to determine an expressive metric for the availability of seed nodes, i.e., the seed density. The absolute number of seed nodes only has low expressive power, since the seed node availability is influenced by additional factors. In particular, the average number of seed nodes available to a simple node is impacted by the deployment area dimensions, the absolute number of seed nodes in the scenario, and the radio range of the seed nodes. This thesis proposes a better metric for the seed density, ρ_{seed}, given in Equation (5.4.1) which considers all of these factors.

$$\rho_{\text{seed}} = \frac{N_{\text{seed}} \times (2r)^2}{A_{sim}}. \tag{5.4.1}$$

The basic idea of the formula is to arrange N_{seed} squares with side length r side-by-side on the simulation area. Each square represents the covered area of one seed node. Although the radio propagation is strongly abstracted in this model, the metric is still much more expressive than only considering absolute numbers of nodes. Larger radio ranges and higher amounts of seed nodes in the network will contribute to the complete coverage of the network and are therefore listed in the numerator. On the other hand, larger deployment area dimensions will make seed coverage of the network much

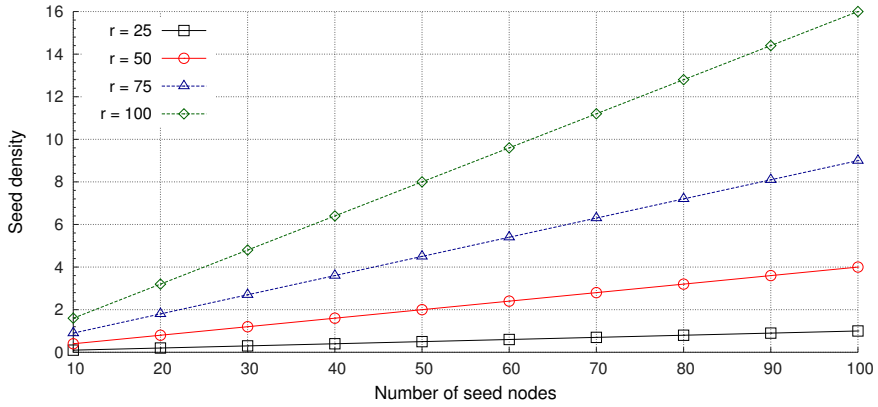


Figure 5.6: Plot of the seed node density function for different radio ranges.

more difficult. Hence, the deployment area is listed in the denominator. Figure 5.6 shows the behavior of the seed node density equation given constant deployment area dimensions of $A_{\text{sim}} = 500\,\text{m} \times 500\,\text{m}$ for different radio ranges. It is obvious that for higher radio ranges the seed node density is growing faster, because higher radio ranges result in better coverage of the deployment area.

The deployment area size and the radio range are kept constant if not stated otherwise. Therefore, it is sufficient to adjust the absolute number of seed nodes to achieve different seed node densities. Table 5.2 shows an overview of the number of seed nodes required to reach different seed node density values given $r = 50$, a total number of 300 nodes in the network and $A_{\text{sim}} = 500\,\text{m} \times 500\,\text{m}$.

ρ_{seed}	0.4	0.8	1.2	1.6	2.0	2.4	2.8	3.2	3.6	4.0
% of all nodes	3.3	6.6	10.0	13.3	16.6	20.0	23.3	26.6	30.00	33.3
absolute no. of seed nodes	10	20	30	40	50	60	70	80	90	100

Table 5.2: Seed node density values for different numbers of seed nodes.

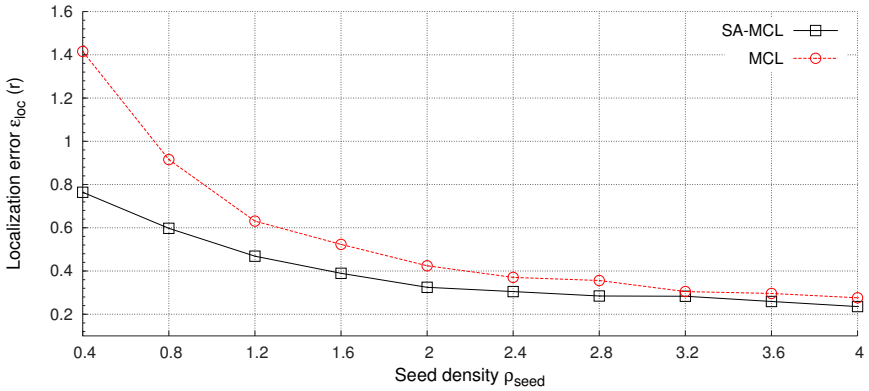


Figure 5.7: Localization error for different seed densities.

5.4.2 Simulation Results

In the following, the effect of every parameter as introduced above is evaluated in detail. The main metric studied is the absolute localization error ϵ_{loc} as explained in Section 2.2.6.

5.4.2.1 Localization Error for Different Seed Densities

The key aim of SA-MCL is to account for situations in which the seed node density is low. Figure 5.7 shows how using the additional sensor information helps SA-MCL to outperform MCL, especially if $\rho_{\text{seed}} < 1$. Here, the number of seed nodes is constantly reduced to achieve lower seed densities. In lower seed density cases SA-MCL ϵ_{loc} decreases by about 40 % when compared with traditional MCL. In cases with higher seed densities there are only few situations left where no seed information is available, i.e., SA-MCL cannot benefit as much from its additional features as in low-density scenarios, leading to a marginal improvement over MCL.

The graph can also be read in a second way. If the main goal is to keep a certain level of localization error, it is possible to reach this level with considerably less seed nodes in SA-MCL compared with MCL. This is an important fact for applications in which nodes are not expected to be recovered after their mission, because the deployment costs can be drastically reduced if it is possible to reduce the number of costly seed nodes.

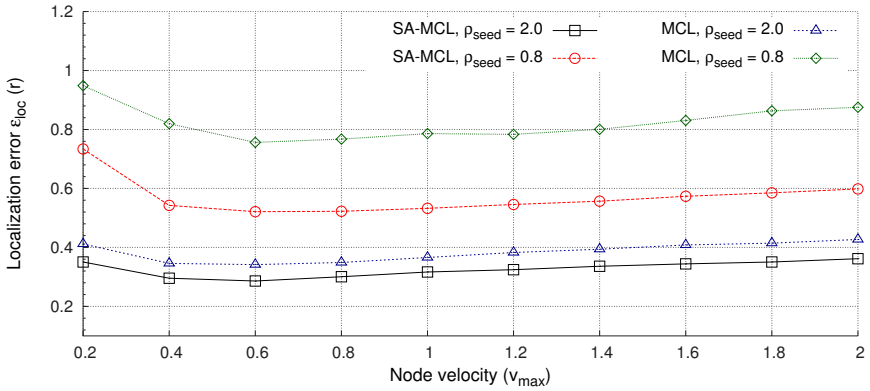


Figure 5.8: Localization error for different node velocities.

5.4.2.2 Localization Error for Different Node Velocities

The localization error for different $v_{\max}$ is presented in Figure 5.8. Up to a $v_{\max}$ value of 0.4 to 0.6 the performance of both MCL and SA-MCL improves continuously. The reason for that is that for lower velocities there is not enough variation in seed node information since the nodes are moving too slow. Consequently, isolated nodes have to wait longer before reestablishing contact to seed nodes again.

After reaching the local minimum, the localization error increases slightly for higher velocities. In these cases, due to more connection losses when moving faster, the nodes cannot gather enough seed node information. However, due to its ability of relying on additional sensor information, SA-MCL outperforms MCL especially if the seed density is low. Even with a high seed node density of $\rho_{\text{seed}} = 2.0$ SA-MCL performs slightly better. However, the difference is much smaller, since situations without seed node information are rare.

5.4.2.3 Localization Error for Varying Sensor Precision

One potential problem in SA-MCL are imprecise sensor readings. All electronic sensors have a limited resolution and might be affected by external impacts. For instance, a magnetometer is always influenced by strong magnetic fields which even pervade possible countermeasures as magnetic shielding.

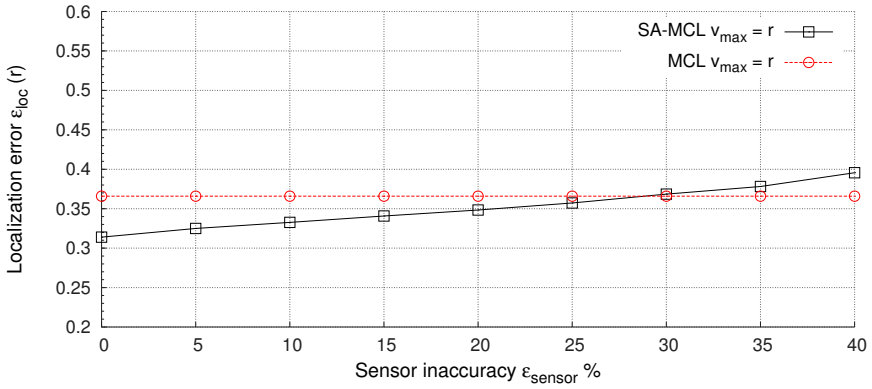


Figure 5.9: Localization error for different sensor precisions.

To account for imprecise hardware, a sensor error is introduced in the simulations ranging from 0 to 40%. The results are shown in Figure 5.9. MCL does not make use of sensor information, therefore the results do not change and are only given for reference. Even if increasing the sensor error up to 30% SA-MCL performs better than MCL. Existing hardware components as used in the field test study presented below in this chapter by far are more precise than the error values assumed for this simulation[5] [144, 145, 146].

5.4.2.4 Effect of Different Sample Set Cardinalities

Several different sample set cardinalities N_{sample} are examined to find a suitable number of samples which need to be maintained for satisfying results. The outcomes are shown in Figure 5.10. Since the sample set cardinality heavily affects the computational overhead of the algorithms, the aim of SA-MCL is to maintain as few samples as possible. For only one maintained sample the localization error is very high, since a single filtered sample results in an empty sample set, which makes a location estimation impossible. As soon as the number of samples is increased the localization error is drastically reduced. In general, for both algorithms it is sufficient to keep a set cardinality of 25 to 30 samples. In accordance with [36], there is no significant improvement after $N_{\text{sample}} = 50$. The reason is that the additional samples do not provide any further information about the location of the

[5] given reasonable calibration

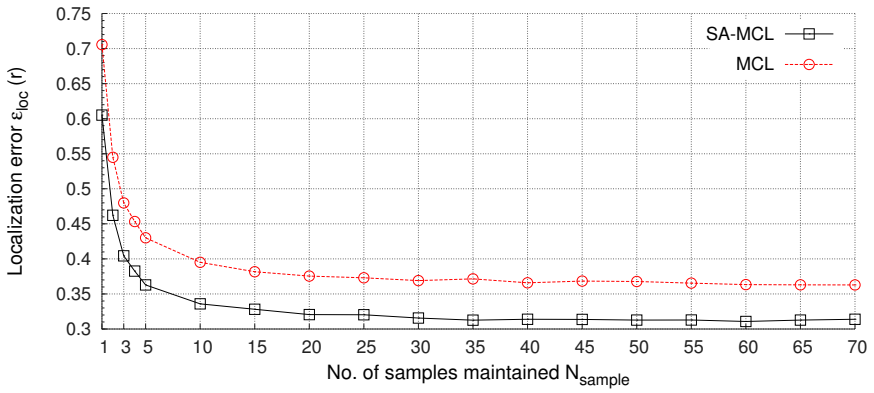


Figure 5.10: Localization error for different sample set sizes.

node. It is advisable to not further increase the number of samples in the set to avoid useless computational overhead.

5.4.2.5 Scalability

An important aspect for algorithms used on devices with restricted computational capabilities is scalability. Scalability describes the ability of an algorithm to handle a growing amount of input while avoiding an increase of computational power necessary to solve the task [148]. An algorithm is considered to be well scaling if it can handle large inputs as efficient as small inputs. In networking algorithms this means that an algorithm working on tiny-sized networks must perform equivalently well if the number of network participants is increased to hundreds or even thousands of nodes.

Fortunately, the concept of range-free localization in association with the broadcast nature of wireless communication allows perfect scalability. In fact, the number of simple nodes a seed node can serve is not limited by computational means. It is of no difference weather a location announcement is received by 1, 5 or 500 different simple nodes. Since the localization algorithm is executed on every single node, no increase of computational resources can be noted. In MCL and its extensions as introduced in this thesis the computational power required is only affected by the number of location announcements received and the sample set cardinality. The aim of any operator of a WSN will be to keep the number of seed nodes as low as

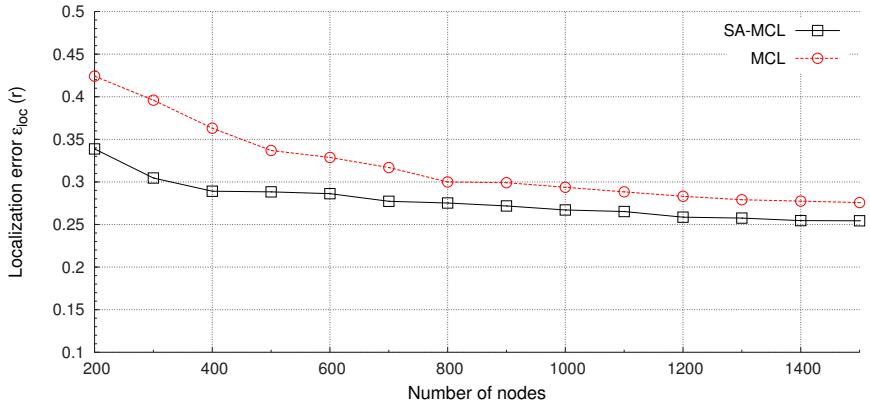


Figure 5.11: Localization error for different amounts of nodes.

possible. As a consequence, it is not likely to have networks with a huge amount of seed nodes. Furthermore, as seen in Section 6.4.2.3, there is no need to increase the sample set cardinality to more than 25-50 samples, as no further decrease of the localization error can be expected. This means both parameters affecting the computational overhead on a node are of no consequence.

To emphasize this Figure 5.11 shows the localization error for different amounts of simple nodes. Interestingly, the localization error is even slightly decreasing for both algorithms in the beginning. The reason for that is the two-hop approach of MCL. Every location announcement of a seed node is repeated by all receiving nodes. With an increased number of total nodes in the network more location announcements will be broadcasted again. Therefore, more nodes which would be isolated otherwise are still receiving location announcements. However, the effect of increasing the amount of nodes in the network is only subtle and reaches a stable level at about 1200 nodes in this scenario. Further increase of the total number of nodes does not have any effect on the localization error.

5.4.2.6 Effect of Different Radio Ranges

Following the definition of the seed density given in Section 5.4 by Equation (5.4.1) the radio range is an important parameter for the seed node coverage in the network. There is a trade-off between the localization error and

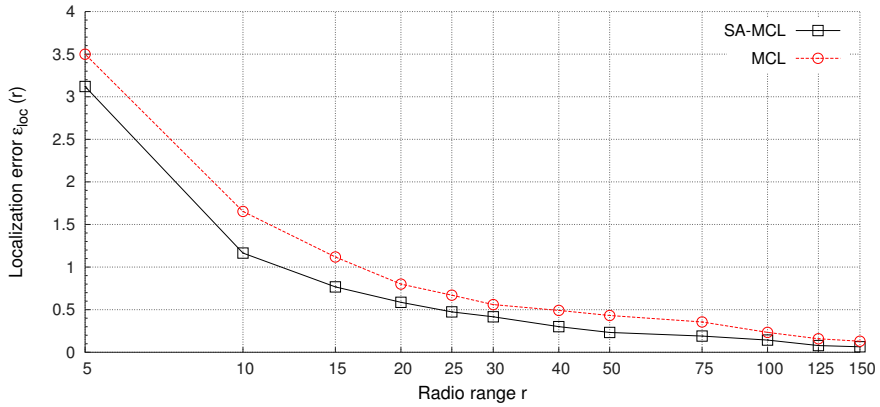


Figure 5.12: Localization error for different radio ranges.

the radio range: the smaller the radio range, the lower the localization error will be given a suitable number of seed nodes to cover the network. Unfortunately, with smaller radio ranges, many more seed nodes are required to cover the whole network and reach reasonable seed densities. Figure 5.12 shows the localization error for MCL and SA-MCL for different radio ranges of both seed nodes and simple nodes. To mainly study the effect of the radio range, in this experiment 200 simple nodes and 100 seed nodes are used, which corresponds to a seed density of 4.0. As expected, for both algorithms the localization error will increase if the radio range increases, since the sample filtering condition is relaxed due to the high radio range.

5.4.2.7 Convergence Time

In Figure 5.13 the convergence time of both algorithms is illustrated. Convergence time denotes the number of simulator iterations until both algorithms reach a stable state, i.e., the initially spread samples concentrate around the simple nodes. The convergence time mainly depends on how fast the simple nodes get contact to seed nodes which in return is mainly dependent on the seed node density ρ_{seed}. As for previously studied parameters, the convergence time is mainly affected by the seed density of the scenario. In this experiment the default values of Table 5.1 are used. While the outcomes confirm the results in [36], SA-MCL shows to behave similar to MCL and reaches a stable state after about 10 iterations. After that, no further improvement is achieved.

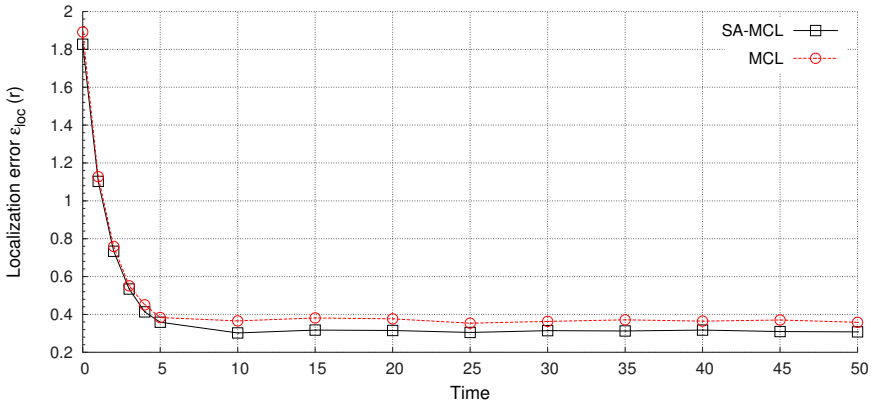


Figure 5.13: Convergence time of MCL and SA-MCL.

5.5 Field Test Evaluation and Results

Simulating algorithms for WSNs is an appropriate option to test and evaluate new ideas and protocols. Furthermore, simulation is often the only possibility to measure important metrics like scalability as thousands of network nodes can easily be emulated with modern software. However, since the real world is only approximated using models for, for instance, antennas, signal propagation, packet loss, and mobility behavior, simulation cannot completely replace field tests and implementations on real hardware. Therefore, in the following, the behavior of both MCL and SA-MCL will be examined on real hardware in a mobile wireless sensor testbed.

Conducting a field test is much more complex, as every node is physically present and parameters which are easy to change in a network simulator (e.g., radio range, battery lifetime, node velocity) are much harder to configure. Even more troublesome are field tests in mobile networks, as implementing mobility in a sensor testbed can be challenging. Although in general it is possible to attach a sensor mote to human beings, it is very inconvenient to keep walking or even running to maintain a continuous level of mobility in the network. A more convenient way to implement mobility is to use radio controlled cars. The mobility behavior in this case is still not representing a real sensor network application and is strongly affected by the driving behavior of the person controlling the car. However, as nodes are moving, topology changes in the network are introduced. In contrast

to simulation, these topology changes are triggered by physical effects and depend on antenna capabilities, environment properties or obstacles.

The field test conducted in this thesis focuses on different aspects. First, a new possibility of developing a mobile WSN testbed using radio controlled cars in general is explored. In addition to that, the testbed is used to implement and evaluate SA-MCL in comparison to MCL. Finally, physical deployment allows analysis of energy consumption, an often stated, but rarely evaluated factor in WSNs.

5.5.1 Used hardware

The following sections describe the hardware components used in the field test. These components include the mobile nodes as well as the added sensors.

5.5.1.1 Radio Controlled Cars

The mobile platform in the field test is a radio controlled car (RC car) named Reely Detonator at the scale of 1:10. The Detonator model was selected because of its capability to master rougher terrain including small stones and wooden sticks while keeping vibrations of the chassis at a low level. Furthermore, it provides enough room on top for all required superstructures. Less sophisticated models in the market mostly use the 35/40 MHz frequency band and therefore have problems if several models are operated at the same time, because the number of radio channels is limited. Consequently, antenna signals of the remote controls may interfere with each other. The consequences are non-moving cars, single remotes driving multiple cars and inability for drivers to properly control their cars. This issue is solved by the RC car, as it uses the 2.4 GHz band and employs *Frequency Hopping Spread Spectrum* (FHHS) or *Direct Sequence Spread Spectrum* (DSSS) both of which avoid colliding antenna signals. Similar to Bluetooth applications a car is exclusively paired with its corresponding remote.

The Detonator is able to achieve velocities of up to 35 km/h. To maintain reasonable velocities for the field test, i.e., to limit the maximum speed of the

vehicle, throttling is required. A simple solution would be to attach a heavy-duty resistor to the circuit in front of the motor. However, these resistors are comparatively expensive and require modifications to the car itself. A better option is thus to slightly change the remote in its behavior. An additional pushbutton replaces the throttle control and the maximum speed can be changed via a potentiometer. Consequently, the pushbutton only provides an on/off functionality and pressing it results in direct acceleration to the maximum velocity defined by the potentiometer. With decreasing battery the test drivers need to readjust the potentiometer to maintain the same velocity level.

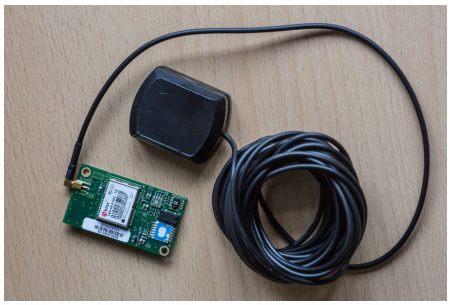
5.5.1.2 Sensor Motes

Two different sensor platforms are used in the field test. The actual implementation of SA-MCL is done on IRIS motes as described in Section 2.1.5.4. The effective radio range of these motes is only about 30 m. To forward live data of the experiment to a base station (i.e., the sink), additional relay nodes are required. These act as static repeaters, which only forward received packets from the mobile nodes to the base station. This part is handled by Advanticsys CM3000 sensor motes [71]. These motes are equipped with larger antennas and therefore reach higher transmission ranges.

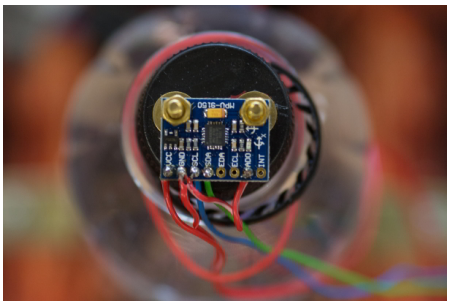
To avoid empty batteries during the field test, the IRIS motes are directly powered via the radio controlled car's batteries. Additional voltage converters are required to transform the 7.2 V provided by the car to the supply voltage of the sensor motes. The relay nodes (CM3000) are not connected to any other power consumers and therefore are battery-powered.

5.5.1.3 Additional Sensors

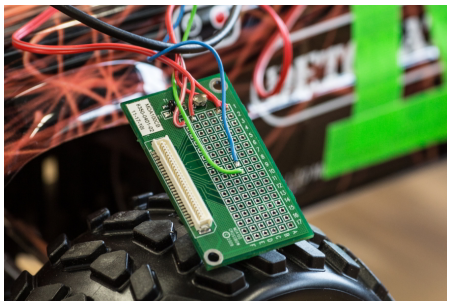
To be able to calculate the localization error, a ground truth reference is required. Although it is theoretically possible to monitor the whole test field area with multiple cameras and set up a hawk eye system as used in professional sports, these systems are very expensive and require precise calibration. Limited manpower and resources render this approach impractical.



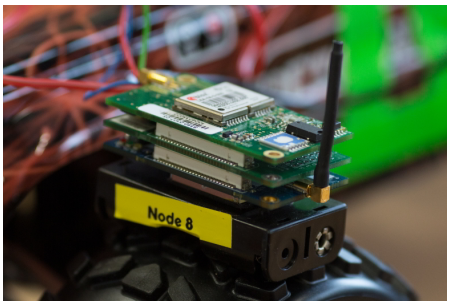
(a) MTS420



(b) MPU9150



(c) MDA100



(d) Sensor Stack

Figure 5.14: Hardware sensors assembled.

Instead, every node is equipped with GPS and uses the positioning information as ground truth. Note that GPS itself introduces its own localization error of up to 3 m [14]. To keep this error at a minimum, the test field has been chosen to provide ideal GPS reception as described below. Crossbow Technologies provides a dedicated GPS module called MTS420 for its MicaZ and IRIS mote product series. Unfortunately the GPS driver provided by the TinyOS community is outdated and does not work at all for the MTS240 GPS boards. Therefore, for this thesis the driver was updated and now works with the provided hardware as shown in Figure 5.14(a). To provide the required additional sensors, an InvenSense MPU9150 [144] board is mounted to the RC cars in the field test. It combines an accelerometer, a gyroscope and a magnetometer on a single chip. Although it is not especially designed to be tiny-sized, the board itself only covers an area of 2 cm $\times$ 1.5 cm, which implies it can be added to many of already deployed sensor motes. Additionally, a *Digital Motion Processor* (DMP) is installed, which is supposed to preprocess the sensor data on hardware level and combines gyroscope and magnetometer information to provide the most precise orientation. Un-

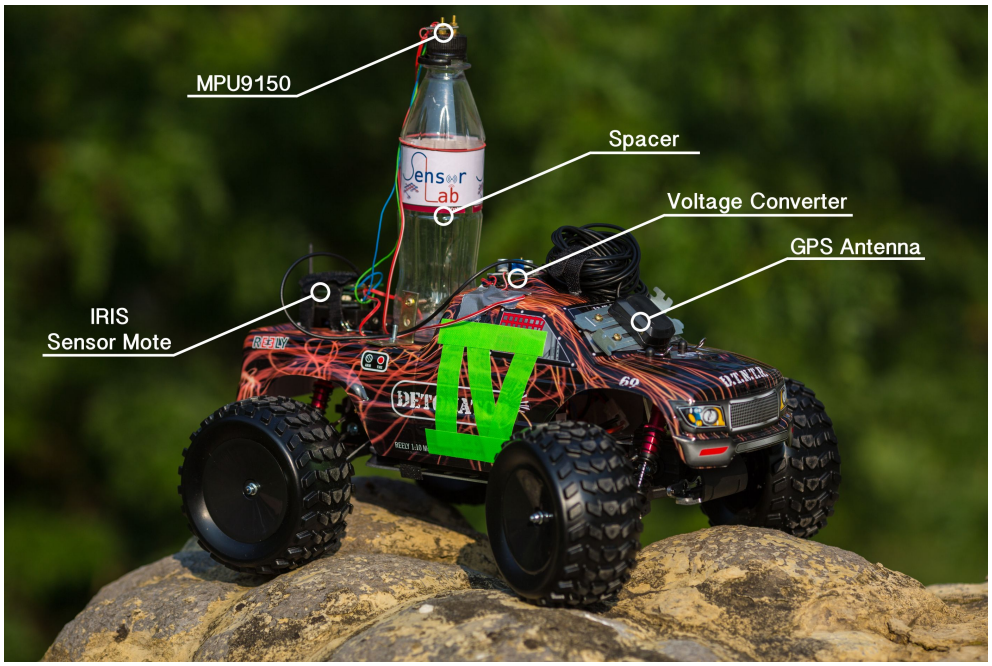


Figure 5.15: Completely assembled RC car.

fortunately, it was not possible to communicate to the DMP using the IRIS sensor motes. Instead, only the raw data of the single sensors can be accessed and has to be processed by the sensor mote itself. The MPU9150 as shown in Figure 5.14(b) is connected to the sensor mote via the expansion board MDA100 (Figure 5.14(c)) manufactured by Crossbow Technologies. The MDA100 provides a large prototyping area for soldering external sensor components. All sensor modules can be put on top of each other and together build the final sensor stack as shown in Figure 5.14(d).

5.5.1.4 Final Car Assembly

Figure 5.15 shows one of the completely assembled test cars. The sensor mote is installed on the flat backside of the car. This ensures the best antenna emission conditions as there are no further blocking parts at this side of the car. To be able to easily remove the sensor motes from the car, while ensuring they are properly secured during the test, the motes are fixed using hook and loop fasteners. The foreside of the car with its additional superstructures introduces signal attenuation and therefore has high impact on the signal

strength. Consequently, the transmission range of the sensor mote is limited in this direction. Initial tests showed that the signal is attenuated by about 25%. This attenuation is unavoidable and thus has to be dealt with in real applications.

At the front of the car the GPS antenna is attached with clear los to the sky. The bottom of the antenna is magnetic. It can be easily fixed to any iron parts. Unfortunately, the whole cover panel of the car is made of plastic. Therefore, two common *Peripheral Component Interconnect* (PCI) slot brackets are screwed on the plastic cover. The GPS antenna is then put on top of these brackets.

The most challenging part is to attach the sensors, i.e., the MPU9150, which SA-MCL requires, to the car. The MPU9150 does not provide any further shielding. Thus, especially the magnetometer is heavily affected by all metal parts of the car. To avoid misreadings, the magnetometer has to be isolated from the rest of the components, while remaining in a fairly horizontal position to avoid tilting. This is achieved by mounting a plastic spacer on the back of the car directly in front of the sensor mote and putting the MPU9150 board on top of that spacer.

The voltage converter as well as loose wires and the power switch (not visible in Figure 5.15) can be held in place using duct tape.

5.5.2 Field Test Limitations and Challenges

5.5.2.1 Hardware Limitations

The original description of SA-MCL as given in 5.2 assumes it is possible to gain more or less exact velocity estimations by integrating accelerometer data over time. Initial tests showed that the accelerometer is too sensitive and even with low pass filtering applied does not provide steady data. Additionally, integrating acceleration to determine the velocity holds only true for uniform acceleration, which is not given by the RC cars. Consequently, a simplified approach similar to MCL is used: Instead of trying to determine the exact velocity of a node it is only distinguished between movement

and no movement and samples are always moved by the maximum possible distance determined by v_{max}.

5.5.2.2 Logistic Challenges

Conducting a field test for a mobile sensor network is incomparably more complicated than in static scenarios. Since every car needs a dedicated driver, a total number of ten network nodes is already challenging to achieve. Although a field test with only ten simple nodes has limited expressive power, it is almost unique in research to conduct such a test in a mobile environment [37].

In addition to the problem of limited human resources, finding a test field, which is large enough to provide a reasonable area, is not easy. To be able to compare the field test results to a simulation with similar parameters, the test field has to be free of obstacles and must provide a flat surface.

5.5.2.3 Mobility Limitations

Mobility in the network is strongly affected by the driver's behavior. Therefore, the resulting network topologies most likely do not directly correspond to any real mobile sensor network application. In addition, the chosen test area described below is comparatively small, as the number of nodes and the radio range of the nodes is also limited. The deployment area of real sensor networks might be considerably larger.

5.5.3 Software Implementation

The implementation is done on IRIS sensor motes [70], which are based on TinyOS and programmed in nesC as described in Section 2.1.5.4. To determine the localization error, distances need to be calculated. In order to simplify these calculations, and to decrease computational overhead on the motes, all recorded coordinates are internally mapped from Gauss-Krüger coordinates to the *Universal Transverse Mercator* (UTM) coordinate

system [81]. The advantage of UTM coordinates is that they can be represented by short integers in contrast to float variables which are required for Gauss-Krüger coordinates. Listing 5.3 shows the transformation of GPS data given in Gauss-Krüger coordinates to the UTM coordinates.

```
event void GpsMsg.newMessage(gps_msg_t *msg, error_t err)
 {
  double e, n;
...
  /* Convert and store raw data to proper latitude/longitute. */
  gps_lat = (float) msg->deg[0] + (((float) msg->minhi[0] + ((
      float) msg->minlo[0] / 100000.0)) / 60.0);
  gps_lon = (float) msg->deg[1] + (((float) msg->minhi[1] + ((
      float) msg->minlo[1] / 100000.0)) / 60.0);
  /* Convert to UTM. */
  UTM(gps_lat, gps_lon, &e, &n);
  /* Center UTM coordinates around experimental field
     and convert to cm. */
  e -= gps_utm_base_e;
  n -= gps_utm_base_n;
  /* Log lat/lon and centered UTM. Also flush log. */
  call Logger.logGPS(gps_lat, gps_lon, e, n);
  call Logger.flush();
...
  /* Otherwise convert to top-left based and store. */
  gps_utm_e = (uint16_t)(24600 + 328 * e);
  gps_utm_n = (uint16_t)(25584 - 328 * n);
}
```

Listing 5.3: GPS data handling.

The sensors provided by the MPU9150 board are connected via the I^2C hardware bus and can be read using APIs supplied by InvenSense. The API is very low level and offers only rudimentary access to the raw data of the sensors. The sensor data is forwarded to the SA-MCL algorithm in two steps. Low level communication using the API of the MPU9150 is used to read the raw data of the sensors. For instance, the magnetometer data is provided as raw float values for each axis. Calculating the yaw angle, i.e. the rotation around the y-axis in a 3d coordinate system, from raw values m_x/m_y can be achieved using Equation (5.5.1). On a flat surface the yaw angle represents the orientation of the node. The corresponding code is shown in Listing 5.4.

$$yaw = \text{atan}(m_x, m_y). \tag{5.5.1}$$

```
//apply calibration data
mx += MAG_BIAS_CORRECTION_X;
my += MAG_BIAS_CORRECTION_Y;
mz += MAG_BIAS_CORRECTION_Z;
//transform raw magnetometer data to orientation in degrees
hdg = 90.0f - atan2(mx, my)*180.0f/M_PI;
```

Listing 5.4: Calculation of car heading.

Although the magnetometer itself provides very good precision, it is vulnerable to tilting and bumps. Both can occur if there are small obstacles, e.g., stones or wood, on the test field. The SA-MCL implementation uses additional data from the gyroscope and accelerometer sensors to mitigate this vulnerability and to stabilize the readings of the magnetometer. This is done by applying low pass filtering to the sensor data to get rid of noise, and by hardening it against tilting by exploiting information collected from the gyroscope.

As described above the accelerometer's main task is to determine if the car is moving or not. This is important to prevent the sample set from being shifted when the car is at rest. Listing 5.5 shows how the shifting vector for the samples is calculated from the determined heading and the constant `MCL_SPEED`, which is corresponding to $v_{\max}$.

```
event void Mpu9150.newData(float hdg, uint8_t flags, float dist)
 {
  /* Log data. */
  call Logger.logHdg(hdg, flags, dist);
  ...
  if ((flags & 1) && hdg <= 360 && hdg >= 0)
  {
   mpu_flags    |= flags;
   mpu_moving   |=   (flags & 1);
   mpu_mag      |= !!(flags & 2);
   mpu_tilt     |= !!(flags & 4);
   mpu_freefall |= !!(flags & 8);
```

```
   ...
  }
  /* If "average angle during movement" value is available,
     calculate vector. */
  if (mpu_count)
  {
   float hdg_rad = (mpu_last_hdg + 270.) * M_PI / 180.;
   mpu_x = (uint16_t)(cos(hdg_rad) * MCL_SPEED);
   mpu_y = (uint16_t)(sin(hdg_rad) * MCL_SPEED);
  }
 }
```

Listing 5.5: Calculation of sample shift vector.

While MCL and SA-MCL share the same code in most cases, they differ once a node is unable to obtain seed information. In this case the code shown in Listing 5.6 is executed. Here, the sample shift vector is added to all samples of the current MCL sample set.

```
  /* If no seeds are detected, rely on sensor assistance. */
  if (!n && (x_off || y_off))
  {
   for (i = 0; i < MCL_N; i++)
   {
    int32_t n_x = (int32_t)L[i].x + x_off;
    int32_t n_y = (int32_t)L[i].y + y_off;

    /* stay within boundaries of deployment area */
    if (n_x >= MCL_COORD_MAX)
     n_x = MCL_COORD_MAX - 1;
    if (n_y >= MCL_COORD_MAX)
     n_y = MCL_COORD_MAX - 1;
    if (n_x < 0)
     n_x = 0;
    if (n_y < 0)
     n_y = 0;
    L[i].x = n_x;
    L[i].y = n_y;
   }
   return;
  }
```

Listing 5.6: Application of sample shift vector.

5.5.4 Field Test Execution Methodology

The following paragraphs describe technical aspects of the field test and how it is carried out in detail.

5.5.4.1 Field Test Area

The test field is a 100 m × 50 m hard pitch for soccer. To record the results of GPS, MCL and SA-MCL, it would be sufficient to use the mobile nodes only, as each node has its data flash memory to store this information. For having a visualization and for the sake of presentation however, it is useful to have a live overview of the current real and estimated positions with respect to the used algorithm. Therefore, every node is sending this information to a central base station as shown in Figure 5.16. The radio range of the mobile nodes is not big enough to have steady contact to the base station. This is solved by using a grid of relay nodes, which are placed at fixed positions in the test field. Their only task is to forward any received packet to the base station. The base station fills the traditional role of the sink in this setup and forwards all received packets to a more powerful laptop computer which will interpret the received positioning information and visualize it by drawing the positions on a background image of the test field taken from Google Maps[6].

[6]Google Maps Service, 2015

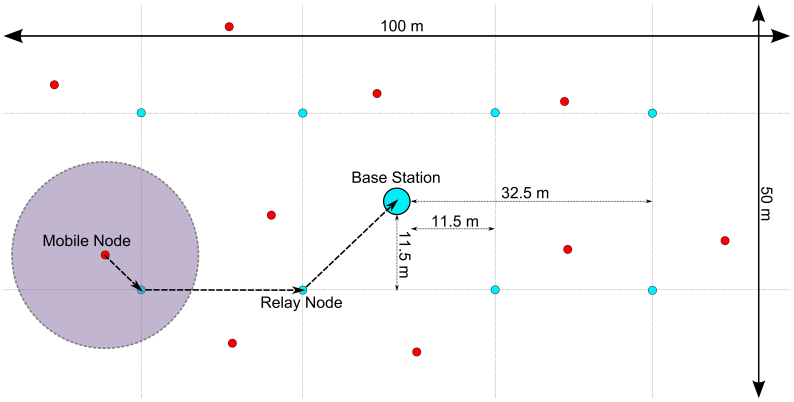


Figure 5.16: General field test setup.

5.5.4.2 Sensor Calibration

Prior to the experiment the car's sensors have to be calibrated. Otherwise the obtained readings might be inaccurate. The calibration process involves leaving the car at rest to allow the gyroscope to settle and thereafter turning the car constantly to allow the magnetometer reaching stable values. In contrast to smart phones the hardware setup used in the field test is neither shielded nor absolutely fixed in its position. Shaking during transportation to the field test area therefore has strong impact on the sensor values and makes calibration unavoidable.

5.5.4.3 Field Test Parameters

The parameters for the field test are chosen based on experience from simulation as well as assumptions about seed node coverage as described in Section 5.4.1. The field test aims to evaluate scenarios where seed information is very rare. Therefore, a low seed density is targeted for the field test to provoke many situations where contact to seed nodes is lost. Since the number of nodes and the deployment area dimensions are fixed, the only remaining parameter is the radio range. Following Equation (5.4.1) a radio range of 5 m, 10 seed nodes and deployment area dimensions of $100\,\mathrm{m} \times 50\,\mathrm{m}$ results in a seed density of $\rho_{\text{seed}} = 0.2$, which is sufficient to achieve high usage of SA-MCL. However, the radio range of the IRIS motes is about six times larger with $\approx$30 m. Lowering the transmission range can usually be achieved by decreasing the transmission power. Unfortunately, the TinyOS version installed on the motes ignores the changed settings and will always send with full transmission power. A workaround is to allow sending with full transmission power while ignoring packets at the receiving node based on the corresponding RSSI. Packets falling short of a certain RSSI threshold are dropped and are not considered for the calculations of (SA-)MCL. It is obvious that this method is less precise than directly limiting the transmission power level. However, radio ranges in real applications are always irregular. The introduced error using the RSSI cutoff approach can therefore be neglected. The radio range of 5 m described above is the lower bound requirement, i.e., the minimum radio range required to achieve the desired

seed density. The effective radio range however is affected by several parameters. For instance, antenna radiation is impacted by the superstructures mounted on the car and drivers walking through the test field can block the direct los, which immediately affects RSSI readings. Table 5.3 lists a rough mapping of RSSI readings to meters based on experimental measurements. However, since a precise mapping of RSSI to real distances is impossible in real world applications (see Chapter 4), Table 5.3 is only presented as an orientation for the reader. The corresponding threshold value for the IRIS motes to achieve a radio range of about 5 m is 29.

The maximum velocity, $v_{\max}$ for the test is set to about 6 km/h. This ensures safe driving without crashes and enables drivers to keep control of their car.

5.5.4.4 Driving Instructions

Sticking close to the simulation setup all test drivers have to follow a sequence of driving instructions, which imitates the random waypoint model.

1. Accelerate the car and drive a straight line
2. Let car roll out and come to a full stop
3. Rest for a short period
4. Change direction and start over

Of course the specific behavior between drivers is different and during the test it cannot be guaranteed every driver is precisely following the driving instructions. For instance, the allure of keeping the car constantly in motion often results in ignored pause times or driving undesired sharped corners. Furthermore, unforeseen events like crashing or tired fingers at the remotes most certainly introduce differences in the mobility behavior compared with arranged simulations. Nevertheless, all these additional impacts are considered to be part of a real world scenario an implementation has to account for.

RSSI	50	33	26	12	9	6
Distance	1 m	4 m	8 m	16 m	22 m	26 m

Table 5.3: Mapping of RSSI values to meters.

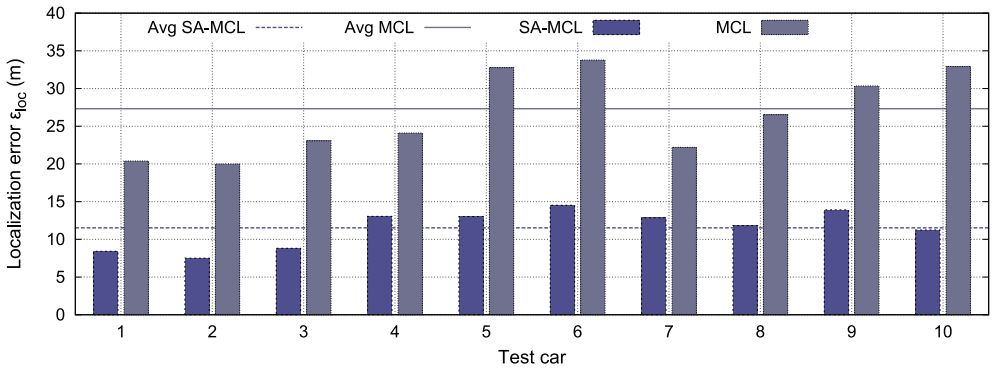


Figure 5.17: Absolute localization errors of MCL and SA-MCL for all test cars.

5.5.5 Experimental Results

After the experiment all logs are collected from the nodes' flash memory. Different error metrics are calculated to evaluate the field performance of both MCL and SA-MCL.

5.5.5.1 Absolute Localization Error

As mentioned in Chapter 2, the key performance indicator of a localization algorithm is the localization error. As stated in Section 2.2.6 the absolute localization error ϵ_{loc} is calculated as the Euclidean distance of two points, namely positions provided via GPS and the estimated locations provided by MCL and SA-MCL. The results for every test car are shown in Figure 5.17. Here, the localization error for SA-MCL is much smaller than the error of MCL. The averages over all cars for both algorithms are drawn as constant lines. Overall, SA-MCL provides a localization error improvement of about 58%. Complementary, Table 5.4 lists the exact results for every car. The superior performance of SA-MCL can be explained due to its ability to account for the missing seed information using its dead reckoning approach. Contact to seed nodes is only occasional, i.e. most of the time SA-MCL has to rely on its collected data information. MCL is lacking this advantage and therefore can update its location estimation only in the rare moments of seed node contact. Due to the continuous mobility, longer periods of seed contact MCL could benefit from are extremely limited.

Car	1	2	3	4	5
ϵ_{loc} MCL	20.39 m	19.99 m	23.06 m	24.10 m	32.77 m
ϵ_{loc} SA-MCL	8.42 m	7.51 m	8.82 m	13.05 m	13.02 m
Improvement	58.71 %	62.43 %	61.75 %	45.85 %	60.27 %
Car	6	7	8	9	10
ϵ_{loc} MCL	33.78 m	26.53 m	30.35 m	32.92 m	29.41 m
ϵ_{loc} SA-MCL	14.53 %	11.83 %	13.91 %	11.24 %	12.96 %
Improvement	56.99 %	55.41 %	54.17 %	65.86 %	55.93 %

Table 5.4: Absolute localization error of MCL and SA-MCL for all test cars.

To reduce the localization error of MCL to the level of SA-MCL, many more seed nodes are required, as the seed density has to be heavily increased. In other words, the same localization error as in MCL can be achieved with a lot less seed nodes in SA-MCL.

5.5.5.2 Grid Localization Error

It is further interesting to see in which regions of the test field in particular SA-MCL is able to outperform MCL. To evaluate this question, a second metric called grid error is presented. Here, the whole deployment area is divided into a grid with a cell size of 3 m $\times$ 3 m. Whenever a node is localized in one of these cells via GPS, the absolute error of MCL and SA-MCL is calculated. The averages of the errors are then plotted in a heat map style for every grid cell.

The grid error is calculated for every car. Averaging all calculated grids results in the final grid error map presented in Figure 5.18. SA-MCL is able to achieve a low localization error on the whole test field, while MCL has problems especially at the outer regions. Cells with an average localization error of up to 90 m can be found, which is even worse than random guessing the current location. The reasons for the bad performance of MCL in these regions is missing seed information. Due to discussed issues of the random waypoint model in Section 2.1.8 and the driving behavior of the test drivers the probability of meeting seed nodes at the outer regions is significantly lower. While MCL has no further possibility to react to these conditions,

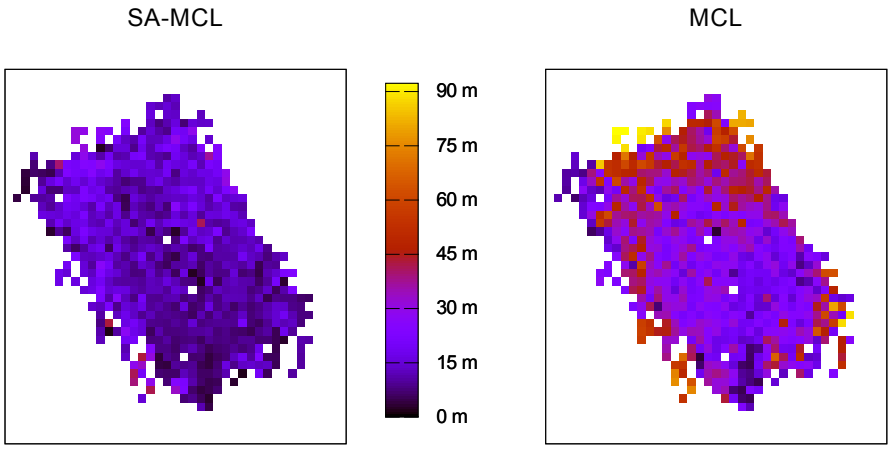


Figure 5.18: Grid localization error averaged over all cars.

SA-MCL again benefits from its additional sensor information. Therefore, the grid error map of SA-MCL looks much more balanced compared with MCL. Confirming the results of the absolute localization error, in case of SA-MCL almost all cells have an average error of less than 15 m.

The individual grid error plots for all cars are provided in the Appendix in Section A.2.1.

5.5.5.3 Optical Trace

By using the log information of all gathered positioning information it is possible to provide an optical comparison of GPS, MCL and SA-MCL. Figure 5.19 shows a snippet of the complete path trace of one of the test cars and provides an optical proof of the advantages of SA-MCL. The car is moving from the top left corner to the bottom of the test field. In this example the contact to all seed nodes is lost, therefore MCL cannot update its position and will start to spread its samples. Consequently, the location of the car is not updated and will jitter around the last known location. Therefore, the path estimated by MCL is only represented by the green spot at the top left corner. As soon as a new location announcement is received at the bottom of the test field, the location is immediately updated, which implies drawing a straight line to connect the two areas. In contrast to that, SA-MCL accounts for the changing orientation of the car and closely imitates the ground truth path provided by the GPS data. Since the discrepancy between the real

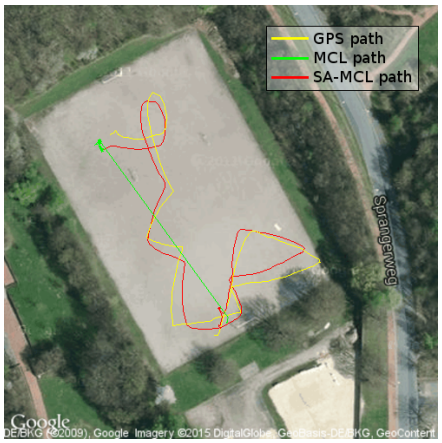


Figure 5.19: Example path trace.

position of the node and the estimation of SA-MCL is much lower, the localization error is much lower, too. The full traces of every car for MCL and SA-MCL are provided in the appendix in Section A.2.2.

5.5.5.4 Energy Consumption

Arguing against GPS usage for localization in WSNs is often based on its high power consumption. In order to confirm this assertion, the current draw of the cars is measured using a MASTECH M9803R multimeter [149]. All components have an active supply voltage of 3.3 V. The results of the current draw analysis are illustrated in Figure 5.20, which shows that the general criticism of GPS regarding its power consumption is reasonable. The MTS420 GPS extension board introduces a current draw of $\approx$ 60 mA which is the majority of all components. The base consumption of the motes is measured while writing log data to the flash memory. It can be assumed that without memory access the current draw is much lower. According to the datasheet of the Atmega1281 [150] processor of the IRIS motes the processor current draw is $\approx$ 7 mA. Current draw introduced by the MPU9150 is comparatively low, i.e., the additional power consumption of SA-MCL is very low compared with its other advantages. In relation to GPS the additional $\approx$ 9 mA of additional current draw is negligible low.

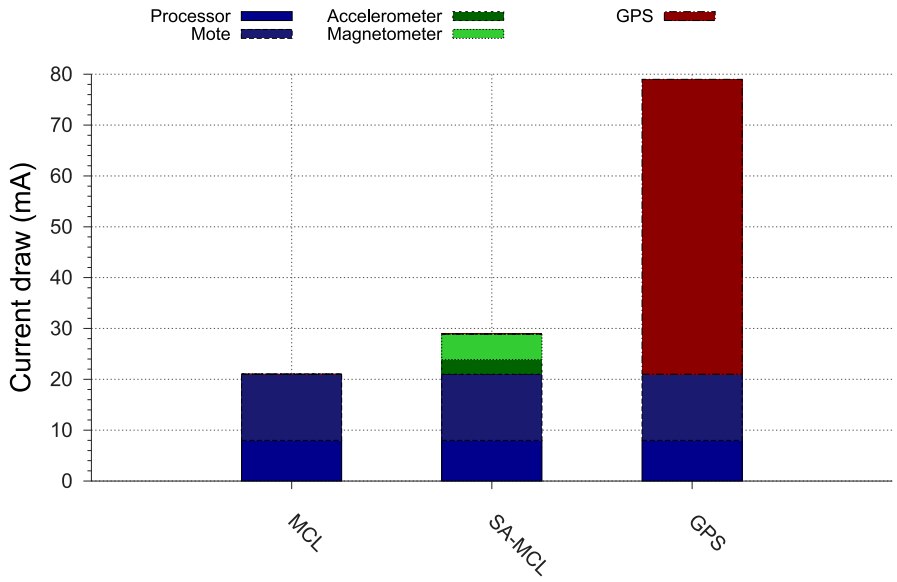


Figure 5.20: Current draw of sensor motes.

5.6 Discussion

5.6.1 Summary

In summary, the presented approach achieves its primary design goal. Using SA-MCL it is possible to account for situations without seed information using the presented dead reckoning method. The seed density definition introduced in this chapter accounts for different deployment area dimensions and radio ranges and therefore has more expressive power than the unknown method used by Hu and Evans to count the number of nodes available on average in each localization approach.

By freezing the state of the sample set and moving it along the traveled path of a node it is possible to reduce the localization error drastically when compared with MCL. However, the precision of SA-MCL mainly depends on the quality of the sensor readings. The evaluation shows that SA-MCL still performs better than MCL up to 30% of sensor error. Given the fact that network operators are interested in reducing the number of seed nodes in the network SA-MCL can maintain a reasonable level of localization error with a considerably less amount of seed nodes than MCL. The simulations show that MCL and SA-MCL both scale very well in large-sized networks and can

even benefit from an increasing number of nodes in the network, as location announcements of seed nodes are forwarded by a larger number of simple nodes. The computational overhead of both algorithms mainly depends on the number of samples maintained by the algorithms and the number of seed nodes in the network. While it is in the interest of the network operator to keep the number of costly seed nodes as low as possible, maintaining a set of 25 samples is sufficient to achieve low localization errors. Therefore, the computational overhead of SA-MCL is negligible low.

The conducted field experiments prove that localization algorithms designed for mobile WSNs can be evaluated on real hardware with reasonable expense. SA-MCL is confirmed to be a feasible solution to bypass the problem of missing seed information. Initial concerns about imprecise sensor information are cleared up as the sensor data is surprisingly accurate.

5.6.2 SA-MCL Limitations

SA-MCL is not designed to completely replace MCL after an initial localization estimation has been conducted. The quality of the location estimation will decrease over time, since sensor errors of magnetometer and accelerometer accumulate. Instead, SA-MCL can be understood as an supporting addition to MCL to bypass situations in which MCL is unable to provide a reasonable location estimate.

The algorithm exploits further sensors to record the movement behavior of a node. Both accelerometer and magnetometer are comparatively cheap sensors, which do not require a lot of space. Therefore, they can be easily integrated in existing systems while keeping additional costs at a reasonable level. However, especially the magnetometer is a sensitive device which requires shielding against outside influences.

In addition to that, the calibration process currently required for each mote as explained in Section 5.5.4.2 is a restricting factor for the deployment process. Future work is required to automated this process to reduce the efforts for the calibration process. It can be expected that with a more sophisticated hardware setup the expenses required for calibration can be drastically reduced as it is the case for hardware in modern smart phones.

Chapter 6

Path-Oriented MCL

In this chapter a new variant of MCL for applications with path-oriented mobility is presented. *Path-Oriented Monte Carlo Localization* (PO-MCL) is designed for nodes which are mainly traveling on a set of paths. These paths are unknown to the nodes at deployment time, but dynamically recognized during operation time. A grid structure representing the deployment area held in the nodes' memory in combination with a magnetometer sensor is used to predict the nodes' movement. With the help of its grid technique combined with an improved sample weighting PO-MCL can reduce the localization error in applications with path-based mobility by about 50 %.

Contents

6.1 Motivation

In many sensor network applications the mobility behavior of nodes differs from the commonly used random waypoint model. Geographical restrictions might detain nodes from accessing certain areas or the mobility behavior of the nodes can be described by more precise models than random movement. Especially the latter is an uprising topic with lots of possible applications in interdisciplinary research [151, 152, 153]. Examples can be found in biology applications like wildlife monitoring as well as in industrial related applications like automated warehouses. In these applications the mobility model can be described as a set of paths which can be expressed as a Graph $G = V \times E$ where E represents the set of single paths and V represents the intersections. In this *path-based* mobility model all nodes in the system are only allowed to move on the edges of the graph. Prominent examples of applications following this model are big cats, the gnu migration, migratory birds [23, 24], insect flight paths, ant trails, cars moving on streets or robots moving between shelves in depots. More recently, biologists and neuroscientists show growing interest in the behavior of flying insects to study mating behavior and group dynamics [154]. Advances in the manufacturing size of modern transceivers as described in Section 2.1.5.4 are going to introduce completely new possibilities of sensing applications, which most likely lead to tiny-scaled sensor networks deployed in insect colonies like hornets or wasps do form.

Given the fact that applications exist in which nodes show the described mobility behavior, it stands to reason to try to exploit it for the localization process. *Path-Oriented Monte Carlo Localization* (PO-MCL) is designed for this specific type of mobility and uses it to achieve a more precise localization in situations where location announcements are not available because of lost contact to seed nodes. Therefore, PO-MCL is also contributing to the problem of missing seed information, as described in Chapter 5.

6.2 Design

The following sections describe the design of PO-MCL. After sketching the main ideas, more detailed information about important aspects of the algorithm is provided.

6.2.1 Main Design Ideas

In PO-MCL all nodes are maintaining a *prediction grid* which divides the whole deployment area into grid cells. A node is always located in precisely one of these cells. A grid cell has exactly eight neighboring cells except for the cells at the borders of the deployment area. Each of the neighboring cells can be labeled with its corresponding cardinal direction (`N`, `NE`, `E`, `SW`, ..., `NW`). A value which represents the probability of moving to this cell next is assigned to all grid cells. Based on observations from seed nodes the grid is updated such that the value of the cell the node has moved to is increased and the values of all other cells are decreased. As long as seed node information is available, the original MCL algorithm is executed, except that samples are assigned the weight of their corresponding grid cell. Therefore samples located in cells where the node currently is or has been before (i.e. cells corresponding to the path the node is moving on) will have a higher weight. In situations without seed information the node relies on the prediction grid information using an initial orientation determined by a magnetometer. The node will try to follow the path on the grid by looking for cells with high values until seed information is available again.

PO-MCL has two additional requirements for each simple node compared with MCL:

- A hardware **magnetometer** used to determine the node's orientation. In contrast to SA-MCL, the precision of the magnetometer is of second rank and is only required to provide a rough heading approximation.
- A **2D array** held in the node's memory to represent the grid structure. The size of the deployment area and the grid cell dimensions determine the additional memory overhead.

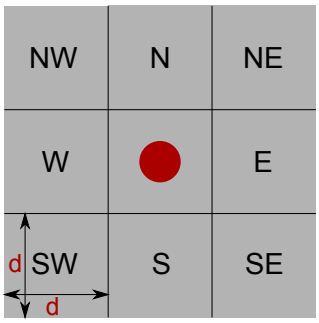


Figure 6.1: Grid directions in PO-MCL

6.2.2 Adaptive Grid Cell Size

The dimensions of the grid cells are an important parameter as they mainly decide over the memory overhead of PO-MCL. The size of the grid is adapted based on the maximum velocity of a node $v_{\max}$ and the localization interval t_{check}, which defines the period between two localization approaches. Since the maximum distance a node can travel between two localization estimations is $d = v_{\max} \times t_{\text{check}}$, the grid cell dimension is also defined as d as shown in in Figure 6.1. It is obvious that for smaller values of d the resolution of the grid is growing and the traveled paths can be mapped to the grid with more precision.

6.2.3 Prediction Grid Construction

In the beginning all grid cells are initialized with the value $0.\overline{1}$, since no information about paths has been gathered yet. Ideally, at every point during operation time of the sensor network the values of all eight neighbors of a cell sum up to 1.0. However, since cells are affecting each other, this is unlikely to happen. In this respect the term probability is not entirely accurate, but used for the ease of presentation.

If a node can update its location estimation based on seed information, i.e., it can execute the MCL algorithm, it checks if it has moved from its previous cell c_{t-1} to a different grid cell c_t. If yes, the probability of c_t is increased, while the probability of the other eight cells including c_{t-1} is decreased. The amount of probability increase Δ_{inc} and decrease Δ_{dec} is determined

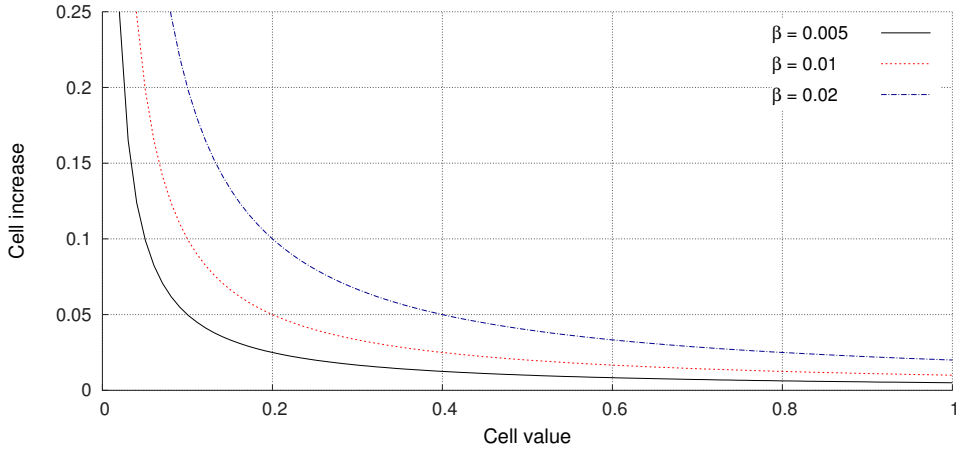


Figure 6.2: Effect of different values for β.

based on the current grid cell value using Equations (6.2.1) and (6.2.2). The parameter β can be used to control the amount of cell increase and decrease. In applications where nodes more likely maintain the same set of paths for the whole deployment time it is desirable to achieve a faster grid conversion, i.e., a larger β should be chosen.

$$\Delta_{\text{inc}} = \frac{\beta}{cellValue} \qquad (6.2.1) \qquad\qquad \Delta_{\text{dec}} = \frac{\Delta_{\text{inc}}}{8} \qquad (6.2.2)$$

Following the definitions of Δ_{inc} and Δ_{dec}, cells with a low probability will be increased faster than cells with high probabilities. To avoid a single cell is constantly increased, a cell can have a maximum probability of 0.5. Furthermore, to provide an upper bound for increasing the cell value Δ_{inc} is limited to be 0.2 at maximum. Otherwise, for cells with very low values (i.e., values $< \beta$) the cell increase Δ_{inc} will become too large. Figure 6.2 shows the effect of different values for β. As explained above, smaller values of β will result in less probability increase and therefore slower grid convergence. An example of updating the grid is given in Figure 6.3. In this example, the node is moving North to c_t, therefore the probability of c_t is increased and the probability of all other cells including of c_{t-1} is decreased.

Over time, the prediction grid will converge to a representation of the traveled paths of the nodes. Figure 6.4 illustrates an example of the convergence process. The figure shows the set of paths, which the grid is supposed to adapt to, and the status of the grid at three different points in time during

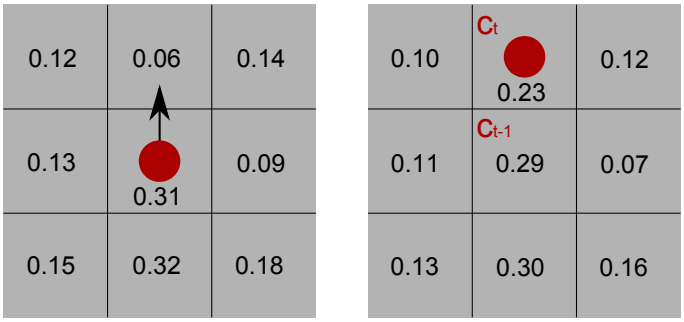
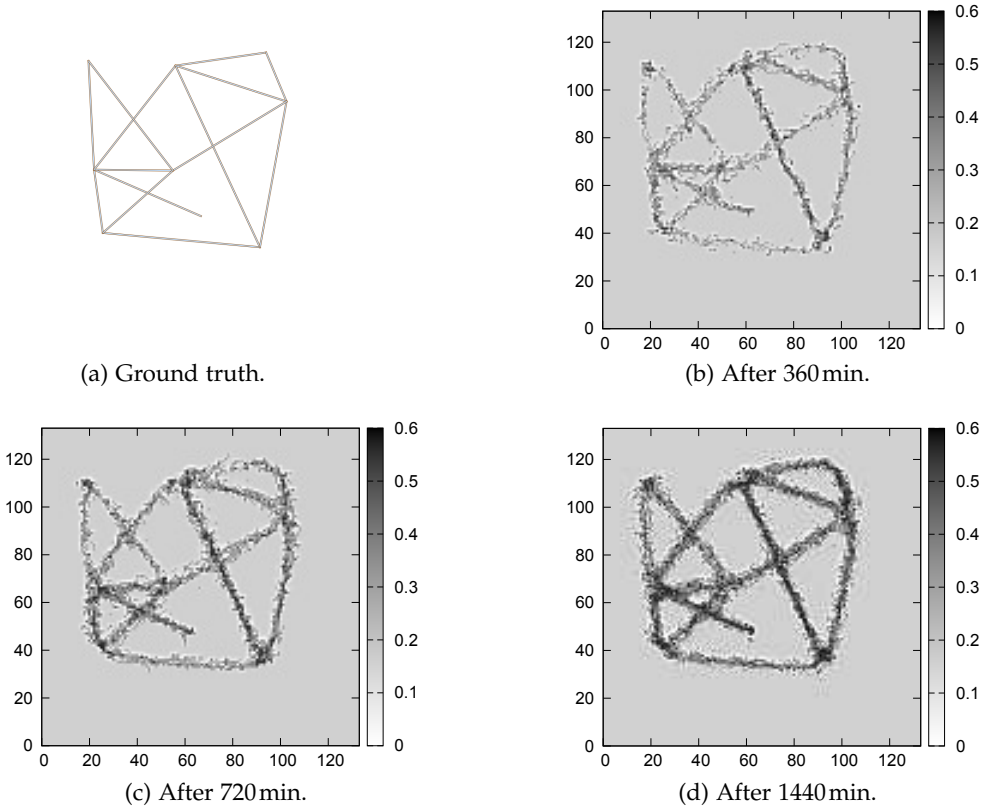


Figure 6.3: Grid update process.

simulation. After 360 min, the path structure is already visible. With ongoing simulation time the grid is converging increasingly to the traveled paths and gives a strongly visible representation after 1440 min.

The convergence time of the grid mainly depends on the seed node density in the scenario. The grid is only updated if contact to a seed node is established. Consequently, the grid is converging faster if seed nodes are present more often. With regard to a long term operation of the network this means that in the beginning more seed nodes should be active to achieve a faster grid convergence. After the paths have been adapted by the grid, it is possible to turn off a fraction of the seeds and to rely on the prediction grid more often instead. Figure 6.4(d) visualizes how the corridors formed on the grid are seamed with white cells which imply that the node will not go past these hems. This is important if the paths are changing during operation time of the network. In this case, old paths might be crossed by new ones in the grid representation, but the hems define clear boundaries which cannot be crossed when using the grid prediction system of PO-MCL as explained in Section 6.2.4. Figures of the convergence process for other path scenarios are listed in Appendix A.3.

The resolution of the grid is mainly affected by the parameters $v_{\max}$ and t_{check}. Figure 6.5 visualizes the grid for different t_{check} and a constant $v_{\max}$ of 5 m/s after 1440 min. If the grid resolution is getting too low, the grid will not be able to map the paths with high enough precision, i.e. the abstraction level of the paths is too high. On the other hand, if the grid resolution is unnecessary high, memory resources will be wasted. Therefore, careful configuration of the localization interval is required.



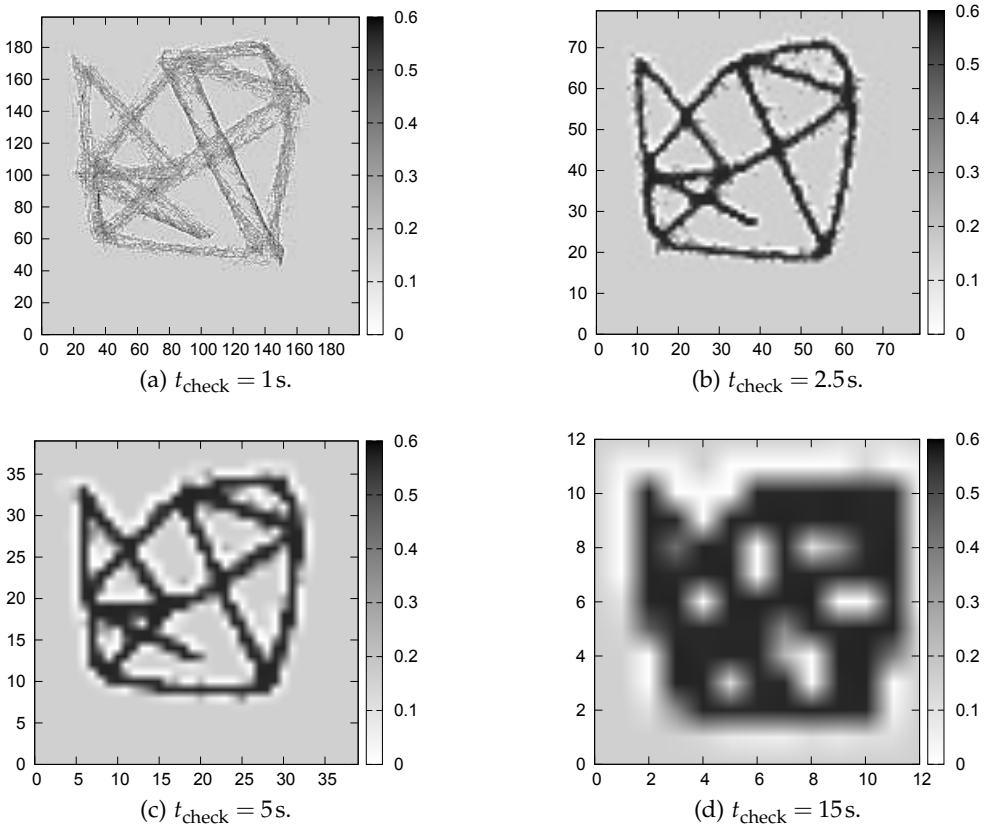
(a) Ground truth.

(b) After 360 min.

(c) After 720 min.

(d) After 1440 min.

Figure 6.4: Grid convergence of a random path scenario.



(a) $t_{\text{check}} = 1\,\text{s}$.

(b) $t_{\text{check}} = 2.5\,\text{s}$.

(c) $t_{\text{check}} = 5\,\text{s}$.

(d) $t_{\text{check}} = 15\,\text{s}$.

Figure 6.5: Different grid resolutions depending on t_{check}.

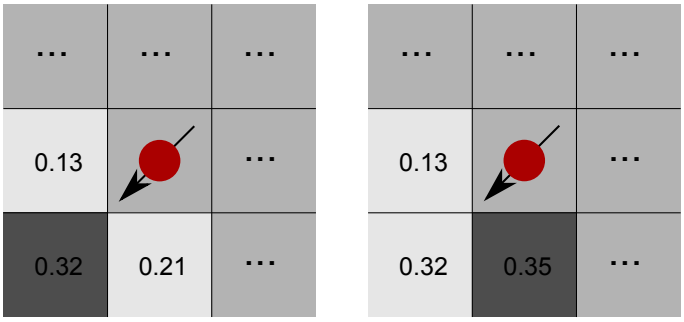


Figure 6.6: Grid movement prediction.

6.2.4 Grid Movement Prediction

In MCL without seed information the sample set L will degenerate gradually over time, as already explained in the beginning of Chapter 5. In PO-MCL the prediction grid can account for these situations. However, without further information in which direction the node is moving to or from which direction it just came, the grid cannot assist in choosing the correct cell for the movement prediction. Therefore, a magnetometer is required to roughly estimate the orientation of the node. The node will then select the three cells in the determined direction and choose the cell with the highest value. All samples of the MCL sample set are then moved by Δd in direction of the determined grid cell. Δd is calculated from the average of the minimum velocity of a node v_{min} and its maximum velocity v_{max} as shown in Equation (6.2.3).

$$\Delta \text{d} = \frac{v_{\text{min}} + v_{\text{max}}}{2} \times t_{\text{check}} \tag{6.2.3}$$

MCL only assigns sample-weights of 1 and 0, depending on if a sample passes the filtering step or not. In PO-MCL sample weights are assigned based on the grid cells where the samples reside in. This attaches more weight to samples residing in cells corresponding to a mapped path. Consequently, for calculating the position estimation Φ_{est} in PO-MCL these samples have more impact.

The grid prediction process is illustrated in Figure 6.6. In this example it is assumed that the magnetometer determined a current direction of `SW` as

indicated by the black arrow. The node selects the three corresponding cells from its prediction grid and looks for the highest value. In the left example the result is consistent with the magnetometer direction (0.32). In the right example the cell in direction `S` has the highest probability (0.35). Therefore the predicted direction of the node indicated by the darker cell is `S` instead of `SW`.

6.2.5 Magnetometer Query Interval

Usage of a magnetometer is required for determining the initial heading of the node as soon as no more seed information is available. Since the magnetometer is consuming additional power it is desirable to use it as little as possible. Hardware tests have shown that the time from powering on the magnetometer sensor to getting a first reading is negligible low ($\approx$10 ms) [144]. Consequently, the magnetometer can be put into sleep mode and will be activated only if required. As the node might change its direction when reaching an intersection of paths, it is necessary to query the magnetometer from time to time to get reliable information about the heading. The magnetometer query interval determines how often this is done. The most precise but also most power consuming solution would be to keep the magnetometer powered on. However, depending on the application scenario and the maximum velocity of a node, it is possible to put the magnetometer to sleep for a couple of localization approaches and fully rely on the prediction grid. How many approaches can be skipped is mainly affected by the application scenario, the time between two localization approaches t_{check} and the maximum velocity v_{max}. In an application where lots of changes in direction (i.e., the path model has lots of intersections) can be expected it is more likely a node will change its orientation more frequently. The faster a node can move, the faster it will reach an intersection. Furthermore, a node might reach an intersection between two localization approaches, which most likely results in a bad grid prediction if the magnetometer is not queried again. Detailed information on the trade-off between the localization error and magnetometer usage is given in Section 6.4.2.6.

```
1: procedure PO-MCL
2:     if |o_t| > 0 then
3:         MCL()
4:         magInterval = 0
5:         updatePredictionGrid()
6:     else
7:         if magInterval % magQuery == 0 then
8:             direction = getDirectionFromMagnetometer()
9:         else
10:            direction = getDirectionFromGrid(posOnGrid)
11:        end if
12:        moveSamplesByDirection(direction)
13:        magInterval++
14:    end if
15:    posOnGrid = determineGridCell()
16: end procedure
```

Figure 6.7: PO-MCL algorithm in pseudo code.

6.2.6 Formal Description

The pseudo code listing of PO-MCL is presented in Figure 6.7. Depending on the fact if the node receives one or more location announcements from seed nodes either the MCL algorithm described in Section 3.2 will be executed or the grid will be used to update the sample set. In the former case the prediction grid is updated if the node has moved to a neighboring cell using the grid update procedure as described in Section 6.2.3. As an additional improvement MCL has been slightly modified to weight samples according to the grid cell values. This is done by assigning the value of the grid cell where a sample is located in to the sample weight, as explained in Section 6.2.4.

The parameter *magQuery* determines how often the magnetometer is checked (e.g., a value of 4 means it es checked every 4th time the node is supposed to localize). *magInterval* is a simple counter to keep track of the number of executed localization attempts in periods where no location announcements are heard. *magInterval* is always reset to zero as soon as MCL can be executed again. After the sample set has been updated in either way, the new grid cell in which the node is located is determined.

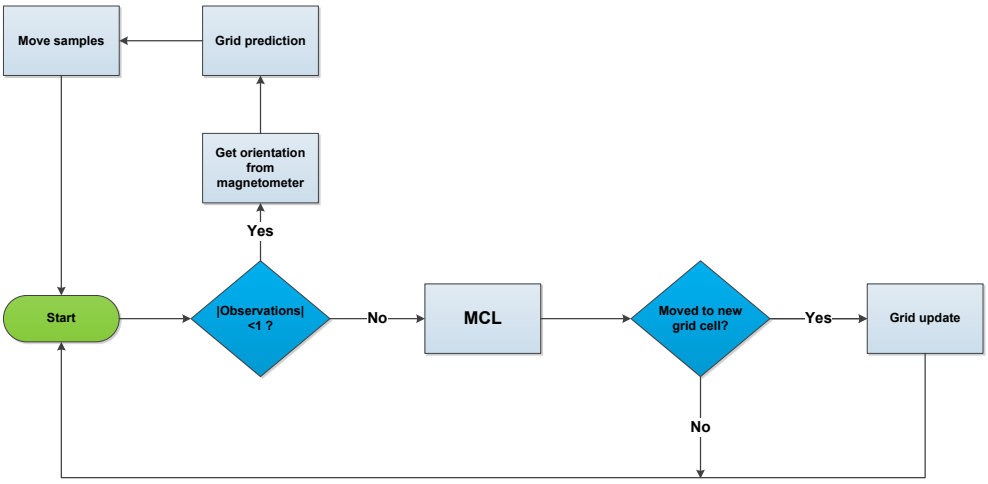


Figure 6.8: Flow diagram of PO-MCL

6.3 Implementation

PO-MCL has been implemented in QualNet [105], a professional network simulation software by Scalable Network Technologies. Information about this simulator is given in Section 2.3.2. To be able to compare PO-MCL with MCL, the Java implementation of MCL provided in [36] has to be ported to QualNet, which is based on the C programming language.

6.3.1 General overview

PO-MCL itself can be put around MCL as shown in the flow diagram in Figure 6.8. At the beginning of the localization procedure it is firstly checked if the node received location announcements. If yes, the MCL branch including the grid update procedure will be executed. If the node is localized in a new grid cell, PO-MCL will update the grid by calculating the new cell values as previously described. In the case that no location announcements are received between two localization intervals the PO-MCL branch is executed. After determining the current orientation using the magnetometer, the grid prediction code is executed to determine the direction the node is most likely moving to. This is used to move the sample set accordingly.

The localization algorithm is implemented as an application layer protocol

(Layer 7 of the ISO/OSI protocol stack, or Layer 5 on the Internet protocol stack, respectively [104]). All application protocols are implemented as a Client/Server-approach in QualNet. Even if communication is assumed to be unidirectional only, the simulator design expects the implementation to be of bidirectional nature. For this reason the function naming convention in QualNet requires the seed nodes to be the server side and the simple nodes to be the client side of the protocol.

QualNet uses a predefined lifecycle for protocols in the simulator. Each model or protocol needs to be registered by including function calls to itself in the QualNet system header files. At the beginning of the simulation every protocol is called to initialize itself. In this part of the code the protocol is supposed to read its configuration parameters, set up necessary variables and reserve required memory. To be able to react on messages sent to the protocol, it has to register an event handler function with QualNet. The protocol code will get called via this function every time a message corresponding to the protocol type is available. In this part of the code the actual program logic is implemented. Finally, at the end of the simulation, a finalization function can be called. It can be used to write out protocol statistic files, to free eventually allocated memory resources and to cleanly exit the protocol code. Listing 6.1 shows the function declarations of the functions registered as QualNet callbacks for PO-MCL.

```
//PO-MCL Seed Node
void AppMCLServerInit(Node *node, unsigned int transmissionRange,
     Address serverAddr, clocktype interval, clocktype startTime,
     clocktype endTime, unsigned tos);
void AppLayerMCLServer(Node *node, Message *msg);
void AppMCLServerFinalize(Node *node, AppInfo* appInfo);

//PO-MCL Simple Node
void AppPMCLClientInitialize(Node *node, unsigned int trRange,
    short noOfParticles, int magFreq, float minSpeed, float
    maxSpeed, clocktype checkInterval, FILTER_TYPE ft, clocktype
    startTime, clocktype endTime);
void AppLayerPMCLClient(Node *node, Message *msg);
void AppPMCLClientFinalize(Node *node, AppInfo* appInfo);
```

Listing 6.1: PO-MCL main functions.

6.3.2 Path-Based Mobility Model

QualNet only provides the random waypoint model, but offers interfaces for integrating own mobility models. Consequently, the path-based mobility behavior the nodes are supposed to show in the simulations has to be added to QualNet first. To support arbitrary path models, the added mobility model features ESRI shapefile input [155]. Shapefiles contain simple vector data for points, lines and polygons and are commonly used in Geoinformatics, e.g., to represent rivers, streets, points of interest or buildings. They are perfectly suited as input files for the path-based mobility model. After parsing the shapefile and building a graph structure from it, a random walk over the graph is computed, which is executed during the simulation. Furthermore, the implementation offers the same configuration possibilities as the random waypoint implementation of QualNet, e.g., choosing a random velocity for path segments from $[v_{\min}, v_{\max}]$.

6.3.3 PO-MCL Seed Nodes

The seed nodes are rather simple to implement, because they only need to send out a location announcement packet in constant intervals. Listing 6.2 provides the C structure representing a location announcement. A location announcement contains the position data of a seed node represented as float values, the unique seed id and a timestamp to discard other instances of the packet when received at the simple nodes. For future extensions of the localization protocol a packet type field in the packet structure has been reserved. Since location announcements are forwarded once by every node, a hop count field is added to the structure which is incremented by every forwarding node.

```
typedef struct {
  unsigned char type;
  float x, y;
  unsigned char id;
  char hopCount;
  clocktype timestamp;
} PMCLAnnouncementPacket;
```

Listing 6.2: PO-MCL announcement packet.

For executing tasks on a regular basis, QualNet provides timers which can be set during the initialization phase of the protocol. If the timer expires, QualNet will generate an event, which must be handled by the protocol code. To be able to distinguish between timers, QualNet uses *timer types* which can be defined by the user. The event message of the expired timer will be sent by QualNet to the function *AppLayerPMCLServer*, which is shown in Listing 6.3. The code provided is responsible for the assembly and broadcasting of a new location announcement. After determining the event type and which timer expired, a new *PMCLAnnouncementPacket* is assembled. The position data can be directly acquired by accessing data structures of the simulator mobility model. In QualNet every node is assigned with an unique ID which can be used as the ID for the announcement packet, too. The timestamp is provided by the simulation clock. After assembly, the packet is spread via broadcast to all adjacent nodes in radio range.

```
void AppLayerPMCLServer(Node *node, Message *msg) {
  char buf[MAX_STRING_LENGTH];
  char clockstring[64];
  AppDataPMCLServer *serverPtr;
  ctoa(getSimTime(node)+getSimStartTime(node), buf);
  switch (msg->eventType) {
   case MSG_APP_TimerExpired: {
   AppTimer *timer = (AppTimer *) MESSAGE_ReturnInfo(msg);
   serverPtr = AppPMCLServerGetPMCLServer(node, timer->sourcePort
       );

   switch (timer->type) {
    case APP_TIMER_SEND_PKT: {
     char *payload;
     PMCLAnnouncementPacket data;
     memset(&data, 0, sizeof(data));
     data.id = node->nodeId;
     data.type = PMCL_PACKETTYPE_ANNOUNCEMENT;
     data.x = (float)node->mobilityData->current->position.
         cartesian.x;
     data.y = (float)node->mobilityData->current->position.
         cartesian.y;
     data.hopCount = 0;
     data.timestamp = getSimTime(node);
     payload = (char *)MEM_malloc(sizeof(data));
```

```
      memcpy(payload, &data, sizeof(data));

      NodeAddress destAddress =
          NetworkIpGetInterfaceBroadcastAddress(node, 0);
      TIME_PrintClockInSecond(getSimTime(node), clockstring);
      APP_UdpSendNewDataWithPriority(node, APP_PMCL_CLIENT,
          NetworkIpGetInterfaceAddress(node, 0), (short) serverPtr
          ->sourcePort, destAddress, ANY_INTERFACE,payload, (int)
          sizeof(data), serverPtr->tos, 0, TRACE_MCL);
      serverPtr->pmclStat.numAnnouncementsSent++;
      AppPMCLServerScheduleNextPkt(node, serverPtr);
      MEM_free(payload);
     }
...
 }
 MESSAGE_Free(node, msg);
}
```

Listing 6.3: PO-MCL seed node code.

6.3.4 PO-MCL Simple Nodes

The simple nodes require more sophisticated code as they implement the essence of the localization algorithm. The following sections show the code for the most important aspects of PO-MCL.

6.3.4.1 Grid Structure

The most important data structure of PO-MCL simple nodes is the grid. The grid is a simple 2D array as shown in Listing 6.4.

```
double** dGrid;
```

Listing 6.4: PO-MCL grid variable.

The grid is initialized in the *AppPMCLClientInitialize* function, which is partly shown in Listing 6.5. The variable *pmclClient* is a pointer to a structure containing all data related to the protocol including the grid. After the number of grid cells has been determined, the memory for the grid is allocated.

```
//calculate distance traveled between two localization
    approaches
pmclClient->maxSpeed = (maxSpeed*checkInterval)/SECOND;
...
pmclClient->maxX = (Int32)dimensions.cartesian.x;
pmclClient->maxY = (Int32)dimensions.cartesian.y;
pmclClient->dCellsX =
((pmclClient->maxX / pmclClient->maxSpeed) + 0.5);
pmclClient->dCellsY =
((pmclClient->maxY / pmclClient->maxSpeed) + 0.5);
pmclClient->current_est_gridx =
(pmclClient->dCellsX / 2.0) + 0.5;
pmclClient->current_est_gridy =
(pmclClient->dCellsY / 2.0) + 0.5;
//allocate memory
pmclClient->dGrid =
(double**) malloc(pmclClient->dCellsY * sizeof(double*));
for(int i=0; i<pmclClient->dCellsX; i++)
 pmclClient->dGrid[i] =
 (double*)malloc(pmclClient->dCellsX*sizeof(double));
//set initial value for every cell
for(int i=0; i<pmclClient->dCellsY; i++) {
 for(int j=0; j<pmclClient->dCellsX; j++)
  pmclClient->dGrid[i][j] = 0.11111;
}
```

Listing 6.5: PO-MCL grid initialization code.

6.3.4.2 Grid Update

Updating the grid is done by firstly determining the cell indexes of the cell the node is moving to depending on the current direction of heading. After some additional checking to ensure the new cell is not out of bounds (i.e. out of the simulation area), the probability increase and decrease, i.e., Δ_{inc} and Δ_{dec} are calculated and applied to all cells. The code for the grid update function is provided in Listing 6.6.

```
void pmcl_updateprobabilities(double** grid, int dimX, int dimY,
    int last_est_gridx, int last_est_gridy, enum direction dir) {
 int d_x, d_y;
```

```
 switch(dir) {
 case NW:
  d_x = -1;
  d_y = -1;
  break;
...
 }
 if(last_est_gridy+d_y > 0 && last_est_gridy+d_y < dimY &&
    last_est_gridx+d_x > 0 && last_est_gridx+d_x < dimX) {
  double currentProbability =
   grid[last_est_gridy+d_y][last_est_gridx+d_x];
  double addedProbability =
   calcNewProbability(currentProbability);
  grid[last_est_gridy+d_y][last_est_gridx+d_x] +=
   addedProbability;
  double d_deltaProb = addedProbability / 8;
  if((-1 != d_x || -1 != d_y) &&
     (last_est_gridx-1) >= 0 && (last_est_gridy-1) >= 0) {
   grid[last_est_gridy-1][last_est_gridx-1] -= d_deltaProb;
   if(grid[last_est_gridy-1][last_est_gridx-1] < 0)
    grid[last_est_gridy-1][last_est_gridx-1] = 0;
  }
...
  if((1 != d_x || 1 != d_y) &&
     (last_est_gridx+1) < dimX && (last_est_gridy+1) < dimY) {
   grid[last_est_gridy+1][last_est_gridx+1] -= d_deltaProb;
   if(grid[last_est_gridy+1][last_est_gridx+1] < 0)
    grid[last_est_gridy+1][last_est_gridx+1] = 0;
  }
 }
}
```

Listing 6.6: PO-MCL grid update code.

6.3.4.3 Location Announcement Forwarding

Announcements from seed nodes are forwarded once by each node in the network to achieve a better provision of seed information. Upon a message from the transport layer is available, QualNet will forward it to the protocol code and trigger an event for it. The received announcement is firstly compared to all already received announcements to eliminate duplicates. Messages from already known seeds but with a newer timestamp are

updated. New announcements are stored in the node's announcement list and forwarded after incrementing the hopcount by sending it to the broadcast address of the network. The code for forwarding an announcement is provided in Listing 6.7.

```
case MSG_APP_FromTransport: {
 UdpToAppRecv *info;
 PMCLAnnouncementPacket* announcement =
 (PMCLAnnouncementPacket*)malloc(sizeof(PMCLAnnouncementPacket));
 info = (UdpToAppRecv *) MESSAGE_ReturnInfo(msg);
 unsigned char* packetData = (unsigned char*)MESSAGE_ReturnPacket
     (msg);
 memcpy(announcement,packetData,sizeof(PMCLAnnouncementPacket));
 free(packetData);
...
 bool bAlreadyReceived = false, bUpdated = false;
 list<PMCLAnnouncementPacket*>::iterator announceIter;
 for(announceIter = clientPtr->listHeardAnnouncements->begin();
     announceIter != clientPtr->listHeardAnnouncements->end();
     announceIter++) {
  PMCLAnnouncementPacket* ap = *announceIter;

  if(ap->id == announcement->id && ap->timestamp == announcement
      ->timestamp) {
   /* duplicate announcement, do nothing */
   bAlreadyReceived = true;
   bUpdated = false;
   break;
  }
  else if(ap->id == announcement->id && ap->timestamp <
      announcement->timestamp) {
   /* known seed, new timestamp -> updated announcement */
   ap->x = announcement->x;
   ap->y = announcement->y;
   ap->timestamp = announcement->timestamp;
   bAlreadyReceived = false;
   bUpdated = true;
   break;
  }
 }
 if(!bAlreadyReceived || bUpdated) { /* save new announcement */
  clientPtr->initialized = true;
  clientPtr->listHeardAnnouncements->push_back(announcement);
```

```
  clientPtr->heardUniqueAnnouncements++;
  if(announcement->hopCount == 0) { /* forward packet */
   char *payload;
   PMCLAnnouncementPacket data;
   memset(&data, 0, sizeof(data));
   memcpy(data, announcement, sizeof(PMCLAnnouncementPacket))
   data.hopCount = 1; /* update hop count */
   payload = (char *)MEM_malloc(sizeof(data));
   memcpy(payload, &data, sizeof(data));
   NodeAddress destAddress =
       NetworkIpGetInterfaceBroadcastAddress(node, 0);
   APP_UdpSendNewDataWithPriority(node, APP_PMCL_CLIENT,
       NetworkIpGetInterfaceAddress(node, 0), (short) clientPtr->
       sourcePort, ANY_DEST, ANY_INTERFACE, payload, (int)sizeof(
       data), APP_DEFAULT_TOS, 0, TRACE_PMCL);
   clientPtr->pmclClientStat->announcementsForwarded++;
   MEM_free(payload);
  }
 }
...
```

Listing 6.7: PO-MCL location announcement forwarding code.

6.3.4.4 Sample Prediction With Grid Weighing

Listing 6.8 shows the code for the sample prediction function including the improved sample weighting of PO-MCL. The sample prediction uses the prediction grid to assign weights to the samples depending on the grid cell the predicted sample resides in. Samples lying on a path will therefore have a higher weight and more impact on the location estimation. Samples are chosen randomly, while making sure the distance between the original sample and the predicted one is smaller than the maximum distance which might have been traveled since the last location estimate.

```
int pmcl_predictParticleGrid(PMCLParticle* p,
                             AppDataPMCLClient* clientPtr) {
 float new_x, new_y;
 float ms = clientPtr->maxSpeed;
 ...
 for(; ;)  {
```

```
  new_x = p->x-ms+(float)(2*ms*(rand()+1)/(RAND_MAX + 2)+0.5);
  new_y = p->y-ms+(float)(2*ms*(rand()+1)/(RAND_MAX + 2)+0.5);

  double distance = distanceOf(new_x, new_y, p->x, p->y);
  if(distance < ms && new_x > 0 && new_y > 0)  {
   p->x = new_x;
   p->y = new_y;
   int xGrid, yGrid;
   /* assign weight from grid cell to sample */
   pmcl_findcellfromcoords(p->x, p->y, clientPtr->dCellsX,
       clientPtr->dCellsY, ms, &xGrid, &yGrid);
   float weight = readGridArray(clientPtr->dGrid, xGrid, yGrid,
       clientPtr->dCellsX, clientPtr->dCellsY);
   p->weight = weight;
   return 0;
  }
 }
 return 0;
}
```

Listing 6.8: PO-MCL grid predict code.

6.3.4.5 Sample Filtering

The sample filtering step is implemented by applying the filter condition described in Section 3.2.2. The function is looping through all received location announcements and checks if the sample passes the filter with respect to the current announcement. First-hop and second-hop seed nodes are checked independently. For samples passing the filter the function returns 1 and 0 otherwise. The corresponding code is provided in Listing 6.9.

```
int pmcl_filterParticle(PMCLParticle* p,
                        AppDataPMCLClient* clientPtr) {
 bool flag = false;
 float tr = clientPtr->transmissionRange;
 list<PMCLAnnouncementPacket*>::iterator announceIter;
 for(announceIter = clientPtr->listHeardAnnouncements->begin();
     announceIter != clientPtr->listHeardAnnouncements->end();
     announceIter++) {
  PMCLAnnouncementPacket* ap = *announceIter;
  if(ap->hopCount == 0) /* check first hop neighbors */ {
   if(distanceOf(p->x, p->y, ap->x, ap->y) < tr)
```

```
    flag = true;
  }
  else if(ap->hopCount == 1) /* check second hop neighbors */ {
   if(tr > distanceOf(p->x, p->y, ap->x, ap->y) &&
      distanceOf(p->x, p->y, ap->x, ap->y) < 2*tr)
    flag = true;
  }
 }
 if(!flag)
  return 0;
 else
  return 1;
}
```

Listing 6.9: PO-MCL sample filtering code.

6.3.4.6 Magnetometer Query Interval

Listing 6.10 shows how the the parameter *magInterval* is used to control how often PO-MCL will use the magnetometer to determine the direction of movement, instead of using the grid. A counter *magCounter* is maintained and increased for every localization approach. In the case that *magCounter* is divisable by *magInterval* the function to get the direction from the magnetometer will be called. Otherwise, the grid is used to predict the node's movement direction.

```
enum direction d;
if(clientPtr->magCounter % clientPtr->magInterval == 0) {
 float realX = node->mobilityData->current->position.cartesian.x;
 float realY = node->mobilityData->current->position.cartesian.y;
 float destX = node->mobilityData->next->position.cartesian.x;
 float destY = node->mobilityData->next->position.cartesian.y;
 d = getDirectionFromMagnetometer(realX, realY, destX, destY);
}
else {
 d = getDirectionFromGrid(clientPtr->lastDir, clientPtr->dGrid,
     clientPtr->current_est_gridx, clientPtr->current_est_gridy,
     clientPtr->dCellsX, clientPtr->dCellsY);
}
clientPtr->lastDir = d;
clientPtr->magCounter++;
```

Listing 6.10: PO-MCL magnetometer frequency code.

6.3.4.7 Magnetometer Emulation

Listing 6.11 shows how the magnetometer is approximated in PO-MCL. As there is no physical sensor present in the simulator, the magnetometer has to be emulated. Using the coordinates provided by the network simulator for its mobility model implementation, it is possible to calculate the current heading of a node. The heading is given in degrees where a value of 0° is representing the cardinal direction East. The common cardinal directions are assigned to their corresponding degree intervals. Consequently, it is possible to inversely map a degree value to its corresponding cardinal direction. For instance, the listing shows that values between 22.5° and 67.5° correspond to the cardinal direction South-East (SE). In case of a calculation problem, i.e., no matching degree interval could be found, a placeholder value *UNKNOWN* is returned.

```
enum direction pmcl_getDirectionFromMagnetometer(float realX,
    float realY, float destX, float destY) {
    //get absolute distance
  float dst_length = distanceOf(realX, realY, destX, destY);
    //convert to degrees
    float dst_direction = acos((destX - realX) / dst_length);
    if (destY > realY) {
        dst_direction = 2 * PI - dst_direction;
    }
    dst_direction = 180*dst_direction/PI;

    //convert to cardinal direction based on degree value
    if((dst_direction >= 0 && dst_direction <= 22.5)||
       (dst_direction >= 0 && dst_direction > 337.5))
        return E;
    else if(dst_direction > 22.5 && dst_direction <= 67.5)
        return SE;
  ... ... ...
    else if(dst_direction > 292.5 && dst_direction <= 337.5)
        return NE;
    else return UNKNOWN;
}
```

Listing 6.11: PO-MCL magnetometer query code.

6.3.4.8 Grid Prediction

The code for predicting the node's movement based on the prediction grid and its last known heading is given in Listing 6.12. As previously described, the node reads the values of the three adjacent cells in direction *currentDir* from the prediction grid. Next, the maximum of these values is determined using the function *maxOfThree*. Based on the returned value the function will return the predicted cardinal direction. The corresponding code is shown in Listing 6.12.

```
enum direction pmcl_getDirectionFromGrid(enum direction
    currentDir, double** dGrid, int currentX, int currentY, int
    xDims, int yDims) {
 double d1, d2, d3, max;
 switch(currentDir) {
 case N:
  d1 =
   readGridArray(dGrid, currentX-1, currentY-1, xDims, yDims);
  d2 =
   readGridArray(dGrid, currentX, currentY-1, xDims, yDims);
  d3 =
   readGridArray(dGrid, currentX+1, currentY-1, xDims, yDims);
  max = maxOfThree(d1, d2, d3);
  if(max == d1)
   return NW;
  else if(max == d2)
   return N;
  else if(max == d3)
   return NE;
  break;
 ... ... ...
 }
}
```

Listing 6.12: PO-MCL grid query code.

6.3.4.9 Sample Shift

To move the samples according to the determined direction, the function *pmcl_moveParticlesByDirection(..)* presented in Listing 6.13 is used. The function is iterating through the node's sample set and adds or subtracts the value of *distance* to or from the x and y coordinates of each sample based on the parameter *dir*, which represents the direction the node is moving in. Additional checking is done to prevent samples from being placed outside of the simulation area. After the sample has been shifted, the value of the grid cell it now resides in is assigned to its weight.

```
void pmcl_moveParticlesByDirection(AppDataPMCLClient* clientPtr,
    enum direction dir, float distance) {
 list<PMCLParticle*>::iterator particleIter;
 for(particleIter = clientPtr->pmclParticles->begin();
     particleIter != clientPtr->pmclParticles->end();
     particleIter++) {
  int x, y;
  PMCLParticle* p = *particleIter;
  switch(dir) {
  case N:
   if(p->y + distance < clientPtr->maxY)
    p->y += distance;
   break;
  case NE:
   if(p->x + distance < clientPtr->maxX && p->y + distance <
       clientPtr->maxY) {
    p->x += distance;
    p->y += distance;
   }
   break;
  }
  ... ... ...
  pmcl_findcellfromcoords(p->x, p->y, clientPtr->dCellsX,
      clientPtr->dCellsY, clientPtr->maxSpeed, &x, &y);
  p->weight = readGridArray(clientPtr->dGrid, x, y, clientPtr->
      dCellsX, clientPtr->dCellsY);
  if(p->weight == 0)
   p->weight = 0.1;
 }
}
```

Listing 6.13: PO-MCL sample shifting code.

6.3.4.10 Calculation of Position Estimation ϕ_{est}

Calculating the final position estimation is done by averaging the contents of the final sample set with regard to the specific sample weight as shown in Listing 6.14. The positions in terms of x and y coordinates are summed up according to the sample weight. Finally, the sums are divided by the sum of all sample weights to retrieve ϕ_{est}. The final coordinates of ϕ_{est} are returned in the variables x and y.

```
void pmcl_estimatePosition(AppDataPMCLClient* clientPtr, unsigned
    int noOfParticles, double *x, double *y) {
 double xSum = 0;
 double ySum = 0;
 double weightSum = 0.0;
 unsigned int c = noOfParticles;
 list<PMCLParticle*>::iterator particleIter;

 for(particleIter = clientPtr->pmclParticles->begin();
    particleIter != clientPtr->pmclParticles->end();
    particleIter++) {
  PMCLParticle* p = *particleIter;
  xSum += (unsigned int)p->x*p->weight;
  ySum += (unsigned int)p->y*p->weight;
  weightSum += p->weight;
 }

 *x = (xSum / (double)weightSum);
 *y = (ySum / (double)weightSum);
}
```

Listing 6.14: PO-MCL position estimation code.

Simulation parameter	Meaning	Default value
$v_{\max}$	Maximum velocity of nodes	2 m/s
t_{check}	Localization interval	2.5 s
N_{sample}	Sample set cardinality	25
$magQuery$	Magnetometer query interval	4
r_{node}	Radio range	50 m

Table 6.1: Simulation default parameters

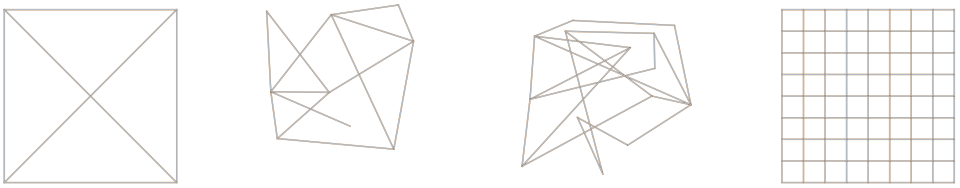
6.4 Simulation Evaluation Setup and Results

PO-MCL is completely evaluated using network simulation. Important algorithmic properties like scalability are difficult to validate without simulation tools, since access to a sensor network consisting of hundreds or even thousands of nodes is usually unavailable. The benefits of network simulation are discussed in detail in Section 2.3.1. Since PO-MCL is implemented in QualNet it is possible to get rid of the abstract unit system, which was necessary to use for the evaluation of SA-MCL in Chapter 5. QualNet features the familiar metric unit system and processes network simulation using a discrete timing event system which allows setting parameters in commonly used units.

6.4.1 Simulation Parameters and Scenario Setup

All experiments are conducted in a deployment area of size 1000 m $\times$ 1000 m. The default parameters unless stated otherwise for all experiments are given in Table 6.1.

Different path scenarios, node velocities, numbers of seed nodes, magnetometer query intervals, radio ranges as well as varying sample set cardinalities are explored. All experiments use 300 simple nodes trying to localize themselves and 25 to 100 seed nodes depending on the studied parameter. Using Equation (5.4.1) given in Chapter 5 and assuming a radio range of 50 m the seed density would be exactly 1.0. However, the equation does not fully apply any longer, since the nodes are not allowed to move arbitrary



(a) Square. (b) RP 1. (c) RP 2. (d) Grid.

Figure 6.9: Input models for different path scenarios.

in the deployment area. Consequently, the probability of a node being at a certain location in the deployment area is no longer given by a uniform distribution. Therefore, in the results section the amount of seed nodes is always given as an absolute value. To provoke a reasonable amount of situations where no seed information has been acquired and to enforce the usage of PO-MCL, it is necessary to use a rather low number of seed nodes.

Every experiment lasts 1 d (1440 min) to provide sufficient time for building the prediction grid. All experiments are repeated 10 times and averaged over all 200 simple nodes to get the final results.

$$\epsilon_{\text{loc}} = \frac{\sum_{i=1}^{N} d(\phi_{\text{real}}, \phi_{\text{est}})}{N} \tag{6.4.1}$$

The localization error is given in multiples of r, as the radio range is the main parameter for determining the absolute localization error (see Section 6.4.2.4). The final localization error ϵ_{loc} is determined by averaging the error of all simple nodes as shown in Equation (6.4.1) where $d(..)$ is the Euclidean distance between the real position ϕ_{real} and the estimated position ϕ_{est} of a node, and N denotes the number of simple nodes.

6.4.2 Simulation Results

In the following, the effect of every parameter as introduced above is evaluated in detail. The main metric studied is the absolute localization error ϵ_{loc} as explained in Section 2.2.6.

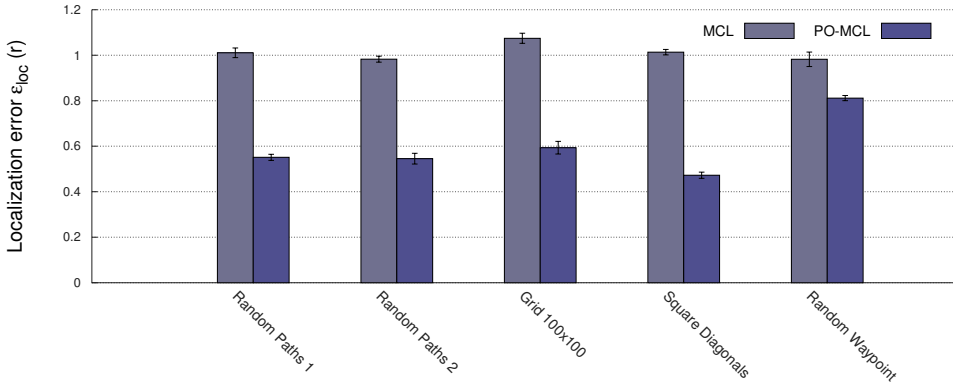


Figure 6.10: Localization error for different path scenarios.

6.4.2.1 Effect of Different Path Characteristics

The effect of four different path characteristics is studied to evaluate how well PO-MCL adapts to these scenarios. A square with diagonals serves as a simple test scenario and provides only four vertices. Therefore, it has only limited expressive power, but can be used as an initial indicator of the performance of PO-MCL. More realistic scenarios are the random path scenarios in which a set of vertices is randomly created and connected using arbitrary edges. The last scenario is a grid with a cell size of $100\,\mathrm{m}^2$ to test the behavior of PO-MCL in situations where a lot of changes in direction can be expected. An overview of the 4 path scenarios tested is given in Figure 6.9.

Figure 6.10 illustrates the localization error for each scenario when executing MCL and PO-MCL. PO-MCL outperforms MCL in all scenarios. Especially in the grid scenario, PO-MCL benefits from using the magnetometer and the prediction grid. On average, the localization error is reduced by about 40 % and even halved in the square scenario. Additionally the behavior of PO-MCL is explored using the random waypoint model as the mobility model. Although PO-MCL cannot efficiently use the prediction grid in this case, it still benefits little from its magnetometer.

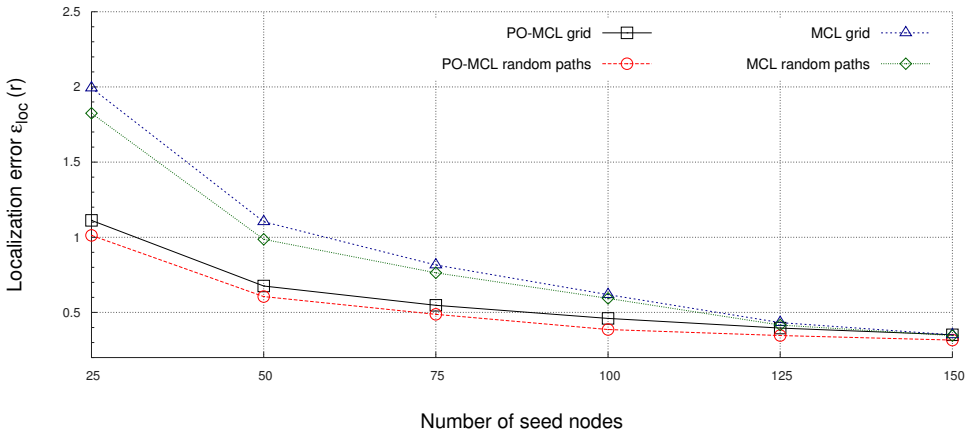


Figure 6.11: Localization error for different number of seed nodes.

6.4.2.2 Effect of Varying Number of Seed Nodes

The most crucial parameter for the overall precision of a localization algorithm is the number of seed nodes available to a simple node on average. In Figure 6.11 visualizes how both MCL and PO-MCL behave if the number of seed nodes in the scenario is constantly reduced. While the localization error for MCL tremendously increases, PO-MCL can compensate the missing location announcements by using the magnetometer and grid prediction techniques. If seed nodes are constantly available, as it is the case for 125 and more seeds, the localization error of MCL and PO-MCL will almost converge to a single curve, although PO-MCL still benefits little from its improved particle weighting.

6.4.2.3 Effect of Different Sample Set Cardinalities

Computational time of both MCL and PO-MCL mainly depends on the number of maintained samples. There is a trade-off between the sample set cardinality and the localization error. It is desired to keep the localization error and the number of samples both as low as possible. Figure 6.12 illustrates that the localization error is rapidly decreasing when increasing the number of samples. This is due to the fact that larger sample sets can account for single imprecise samples. Using too few samples results in a large localization error, since most of the time not a single sample fulfills the filter condition, i.e., the filtered sample set is empty. However, after a sample set cardinality of 25 is reached, there is no further improvement of the localization error.

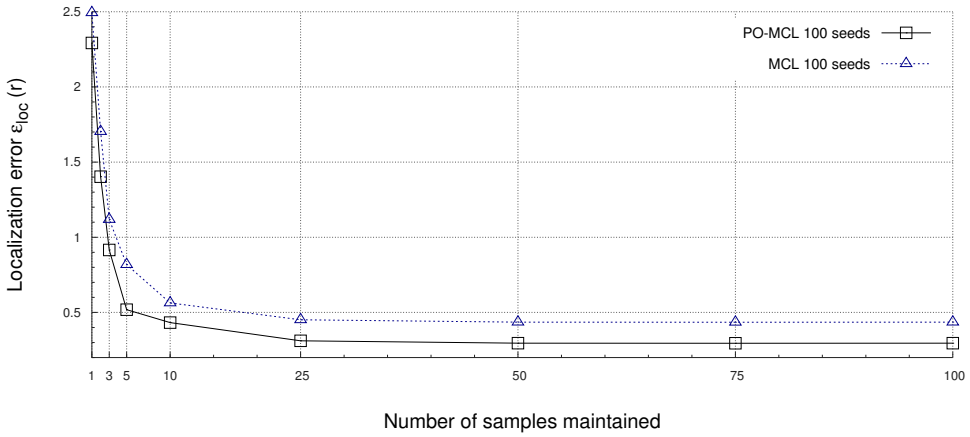


Figure 6.12: Localization error for different sample set cardinalities.

6.4.2.4 Effect of Different Radio Ranges

Since MCL and PO-MCL both are connectivity-based algorithms the absolute localization error is mainly determined by the radio range r of the nodes. Smaller values of r will result in smaller absolute localization error given that a sufficient number of seed nodes is available. On the other hand with smaller r a bigger number of seed nodes is required to ensure the same level of seed node coverage. Figure 6.13 shows the absolute localization error when increasing the radio range. Due to less seed coverage, the localization is large for small radio ranges of 5 m-10 m and decreasing rapidly when increasing the radio range to about 25 m-50 m. In contrast to the evaluation study of SA-MCL, in PO-MCL further increase of the radio range leads to an increased localization error as well. The reason is a growing risk of packet collisions with increased radio range. Since all seed nodes send their location announcements exactly at the same time, the probability of colliding packets is very high. However, this problem could be fixed easily by applying a jittering mechanism, i.e., each seed node will wait for a random period before sending the location announcement.

6.4.2.5 Effect of Different Node Velocities

In Figure 6.14 different node velocities are investigated. The results are showing the typical behavior of MCL as it is explained in Chapter 5. Both al-

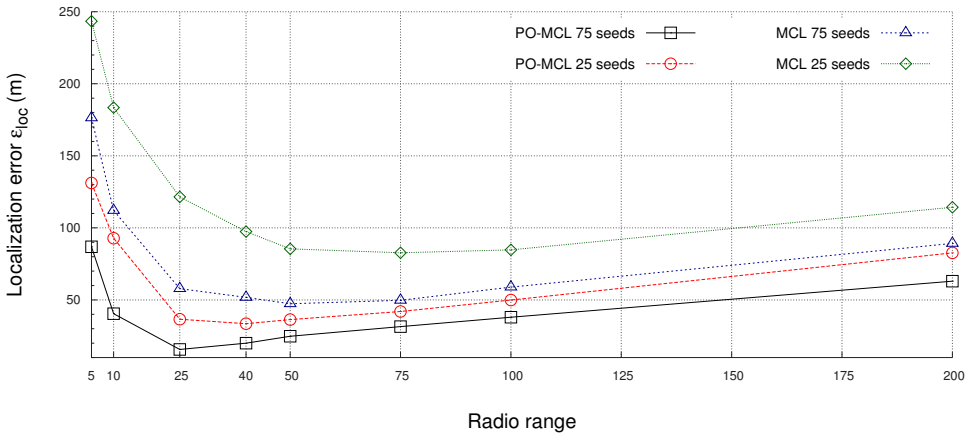


Figure 6.13: Localization error for different radio ranges.

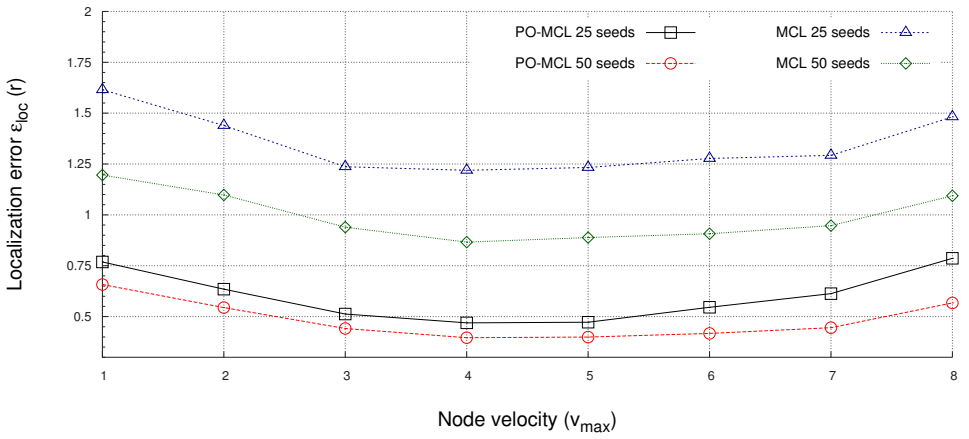


Figure 6.14: Localization error for different node velocities.

gorithms benefit from an increasing node velocity in the beginning, since periods without seed information are getting shorter for faster moving nodes. However, since the radio range is kept the same, for higher node velocities of $>4\mathrm{m/s}$ the localization error is increasing as nodes lose contact to seed nodes more often. Depending on the radio range the local minimum of the curve might be found at a different node velocity, but the characteristics of the curve will be the same for other simulation parameters.

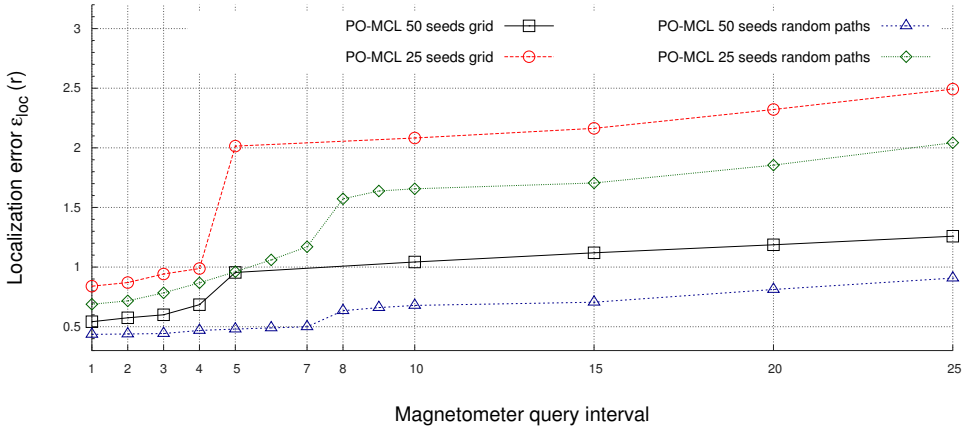


Figure 6.15: Localization error for different magnetometer query intervals.

6.4.2.6 Effect of Different Magnetometer Query Intervals

A very important parameter for the power consumption of PO-MCL is the magnetometer query interval, which describes how often PO-MCL will use the magnetometer to determine its current direction of movement. There is no general answer to the question how often the magnetometer needs to be queried to maintain a low localization error, since this mainly depends on the application scenario. Precisely, in scenarios with lots of intersections a node most likely is going to change its direction of movement more frequently. Consequently, the magnetometer should be queried more often to avoid missing changes of orientation. In contrast, in scenarios with long path segments the magnetometer query interval can be relaxed as not many changes of direction can be expected.

Figure 6.15 presents the results of different magnetometer query intervals. Obviously, the best results are achieved when the magnetometer is active all the time. In this case, the grid is only used for sample weighting and not for predicting the movement of a node. When increasing the magnetometer query interval up to values of 3 to 4, only slight increase of the localization error is noted. Especially for the grid scenario, higher values result in increase of the error, because changes of direction happen more often than detected by the magnetometer. The characteristics of the curve heavily depend on the other parameters. If t_{check} is decreased, whereas keeping the same v_{max} PO-MCL will be executed in shorter intervals and therefore the magne-

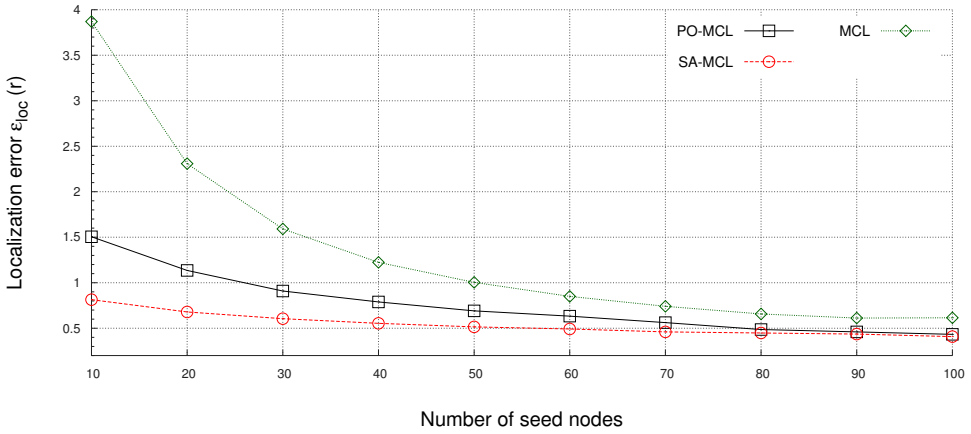


Figure 6.16: Comparison of PO-MCL and SA-MCL with random waypoint mobility.

tometer will be queried more often, while the same distance is traveled by a node. Consequently, for smaller values of t_{check} longer magnetometer query intervals are possible.

6.4.2.7 Comparison with SA-MCL

PO-MCL is especially designed for applications in which path-based mobility can be assumed, while SA-MCL can always rely on its sensor information no matter what kind of mobility model underlies. The advantage of PO-MCL is that additional sensor information is required less frequent compared with SA-MCL where the magnetometer must constantly supply new orientation values. In PO-MCL this is compensated by using the grid mapping of the paths as a reference for the node's movement. Which method is most suitable depends on the specific application. SA-MCL is the more general approach, which has been proven to be robust against missing seed information. However, in very energy-critical networks with path-based mobility PO-MCL might be the better option, as it will consume less additional power, because the magnetometer is allowed to be turned off more often.

In Section 6.4.2.1 it is already shown how PO-MCL is performing when applied to a random waypoint scenario. Figure 6.16 extends these results by showing the results when compared with SA-MCL in a random waypoint

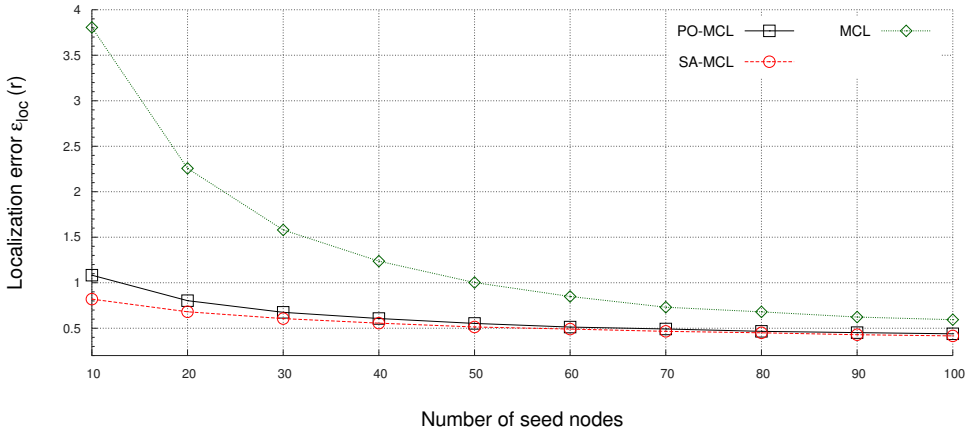


Figure 6.17: Comparison of PO-MCL and SA-MCL with path-based mobility.

scenario for different seed node densities. Both SA-MCL and PO-MCL have a huge performance gain compared with MCL. However, PO-MCL cannot make use of its grid prediction method since there are no paths in the scenario the grid could converge to. Consequently, its localization error is still clearly larger compared with SA-MCL.

Figure 6.17 illustrates how the results will change if a path mobility scenario is examined. Here, PO-MCL can make use of its grid prediction mechanism and achieve better results. However, SA-MCL is still performing slightly better, since the sensor information is more precise than the grid approximation of the traveled paths.

6.5 Discussion

6.5.1 Summary

This chapter presented PO-MCL, an approach to exploit node mobility behavior to achieve a lower localization error for applications where nodes are mainly moving on a set of paths unknown to the sensor node in the beginning of the network operation. The grid technique of PO-MCL is able to create a representation of the traveled paths of a node. Tn combination with a magnetometer the grid is used as an additional source of information for predicting the nodes direction of movement in situations where no seed information is available. Furthermore, the grid allows a more sophisticated

particle weighting which further reduces the localization error. The magnetometer query interval has crucial impact on the performance of PO-MCL and needs to be chosen adequately depending on the expected number of changes in direction.

The evaluation of PO-MCL indicates the advantages of the grid technique. The localization error can be reduced by 40 %-50 % in all tested path scenarios. Even if the random waypoint model is applied, PO-MCL still benefits slightly from its initial usage of the magnetometer to determine the initial node heading. Acknowledging the results of Hu and Evans for MCL [36] and the observations of Chapter 5 PO-MCL shows the same curve characteristics when evaluating different node velocities, sample set cardinalities and seed node amounts.

6.5.2 Limitations

PO-MCL is designed for applications in which nodes tend to move mainly on a finite set of paths. Although there is a small advantage of PO-MCL in comparison with MCL when applying both in a random waypoint scenario, the additional cost bears no relation to the slightly reduced localization error. Therefore, PO-MCL is only suitable for a certain type of applications in mobile WSNs, which show the required mobility behavior.

The ability to use the grid as a source for movement prediction is strongly depending on the grid resolution. In Section 6.2.3 it is described that the cell dimensions are derived from the maximum velocity of a node and the localization interval. To achieve a more detailed resolution, the localization interval needs to be shortened which will result in additional computational overhead and increased memory consumption. The grid resolution must match the number of paths in a scenario, i.e., the more paths can be expected the finer the resolution of the grid needs to be. Otherwise the grid representation will be too abstract and merge distinct paths together. For future improvements of the protocol it might be worth considering the radio range as an additional parameter to determine the grid cell dimensions. In so far fictional applications like sensor networks in ant colonies the radio range will be drastically reduced, while the paths of the nodes might require a very high grid resolution.

Chapter 7

Conclusion

In this last chapter, the thesis and its contributions are summarized. Beyond that, possible future research items, which extend or refine the results and methods presented in this thesis, are stated.

Contents

7.1 Summary

Localization is an ongoing topic in sensor networks research. Expanded demands like node mobility and decreasing manufacturing size lead to new challenges and require improved solutions for efficient localization techniques.

This thesis initially analyzed RSSI, an often used ranging technique, regarding its suitability for the usage in range-based localization algorithms in Chapter 4. Several possible factors of impact were listed including physical hardware capabilities, theoretical problems of mapping RSSI readings to distances, and signal attenuation. Detailed measurements were performed to analyze the RSSI behavior. Although the basic assumption of a decreasing RSSI with growing distance holds true, a useful mapping to absolute distances is not possible due to signal unsteadiness. Reasonable usage of RSSI is only possible in controlled environments as performed in previous studies. However, these scenarios require calibration, fixed sensor positions and clear los. In rapidly deployed or mobile networks neither of these requirements can be fulfilled. Therefore, robust localization applications cannot rely on active ranging based on RSSI. The outcomes of this study led to focusing on improving range-free localization techniques.

One of the most promising proposals for range-free localization is the Monte Carlo Localization approach. Compared to solutions like APIT or Centroid, which require at least three present anchor nodes, in MCL one anchor node is sufficient to give a first location estimate, although the precision is increased with two or more anchors. Furthermore, MCL is one of the first approaches, which accounts for full node mobility. The main problem in mobile sensor networks are isolated nodes, which temporary lose contact to all anchor nodes. Chapter 5 described the effect of the degenerating sample set of MCL for isolated nodes and proposed a countermeasure based on a dead reckoning approach called *Sensor-Assisted Monte Carlo Localization* (SA-MCL). In the event of losing contact to all anchor nodes, a node will use common ots sensors to determine its velocity and heading. Instead of executing the prediction step of MCL, which would lead to sample set degeneration, in SA-MCL the node will move its samples along the path it travels relative to its last estimated position. Extensive simulation was

presented to show that SA-MCL is able to account for missing seed information. The results indicate that SA-MCL can reduce the localization error drastically by up to 50% depending on the amount of seed nodes present in the scenario. Important parameters like radio range, node velocity and number of nodes were evaluated to study the behavior in terms of scalability and mobility effects.

After the simulation studies, SA-MCL was implemented on real hardware and evaluated in a small mobile sensor network testbed. IRIS sensor motes mounted on radio controlled cars prove to be a feasible solution to account for mobility in sensor network testbeds. The results of the field test evaluation showed that SA-MCL indeed has superior performance compared to MCL as the localization error in this scenario could be reduced by about 60%.

Mobility in sensor networks has often been considered to be a challenge only, instead of a possibility for new localization solutions. In Chapter 6 it was shown how specific mobility behavior can be exploited for the usage in localization. In *Path-Oriented Monte Carlo Localization* (PO-MCL) it is assumed all nodes in the network only travel on a set of paths. These paths are unknown to the nodes a priori and are mapped to a grid each node has to maintain during network operation time. The grid is updated whenever a node executes MCL to estimate its location. It was shown that over time, the grid converges to the paths the nodes are traveling on. In the absence of seed nodes a node can use the grid to predict its next movement direction and use it for a more accurate location estimate. Furthermore, PO-MCL introduces a more sophisticated sample weighting based on the prediction grid to strengthen the impact of samples residing in grid cells corresponding to the traveled paths. Similar to SA-MCL, a node will move its samples in direction of the predicted direction, instead of executing the MCL prediction step. To determine the initial heading of a node, a magnetometer is used. Detailed simulations showed that it is not necessary to query the magnetometer every time the localization algorithm is executed. Depending on the density of paths and intersections a threshold value for every scenario can be found, which determines the magnetometer query interval. Compared to MCL the localization error is drastically reduced in scenarios with path-based mobility by about 50%, depending on the seed node density and the

application scenario. In direct comparison to SA-MCL in a path-based mobility scenario PO-MCL shows a slightly worse performance, which means SA-MCL is the more precise solution. However, PO-MCL requires less additional sensor information and does not need to query its magnetometer all the time and therefore is the more energy efficient approach. Depending on the application scenario both approaches have their raison d'être.

7.2 Outlook

Due to newly raising challenges in sensor networks research, the development of new solutions for localization is never completed. Advances in manufacturing size and energy consumption lead to tiny-sized sensor motes, which will allow completely new areas of application. However, efficient algorithms in terms of computational efficiency and energy consumption are required to meet the requirements of these applications. The research conducted in this thesis mainly focuses to improve the localization error and to reduce deployment costs by decreasing the number of required seed nodes. Although these goals are achieved, the absolute localization error is still quite large compared to GNSS solutions like GPS. Future research is required to further decrease the error, while maintaining the same resource overhead.

For SA-MCL it is desirable to perform more field tests with different parameters considering different radio ranges and higher node velocities. Furthermore, the built testbed can also be used for other experiments. For instance, to confirm the simulation results achieved for PO-MCL, a field test similar to the one conducted for SA-MCL needs to be performed. While some of the components built for SA-MCL can be reused, others need further investigation. Especially the prediction grid, which requires a comparatively large amount of memory, could exceed the currently given hardware capabilities.

Another important factor for the success of every deployed sensor network is security. In this thesis no malicious nodes, which could interfere with or even disable the localization process, are considered. For instance, the classic wormhole attack [156] could be used to replay location announcements at physically impossible locations. Consequently, without detection a node

receiving these announcements would localize itself at a different position than usual. Even more drastically fake announcements sent by bogus nodes can be used to determine the position where the node will localize. This can have mission critical consequences and needs to be prevented. Future research has to develop techniques to detect fake announcements. Several countermeasures have already been developed for routing in sensor networks [157, 158, 159], which is subject to the same problem.

Bibliography

[1] W. W. Dargie and C. Poellabauer, *Fundamentals Of Wireless Sensor Networks: Theory and Practice*, 1st ed. Wiley, 2010.

[2] Z. Pei, Z. Deng, B. Yang, and X. Cheng, "Application-oriented wireless sensor network communication protocols and hardware platforms: A survey," in *IEEE International Conference on Industrial Technology, 2008. ICIT 2008.*, April 2008, pp. 1–6.

[3] M. Taleghan, A. Taherkordi, M. Sharifi, and T. hoon Kim, "A Survey of System Software for Wireless Sensor Networks," in *Future Generation Communication and Networking (FGCN 2007)*, vol. 2, Dec 2007, pp. 402–407.

[4] M. Di Francesco, S. K. Das, and G. Anastasi, "Data Collection in Wireless Sensor Networks with Mobile Elements: A Survey," *ACM Trans. Sen. Netw.*, vol. 8, no. 1, pp. 7:1–7:31, Aug. 2011. [Online]. Available: http://doi.acm.org/10.1145/1993042.1993049

[5] S. Isik, M. Y. Donmez, C. Tunca, and C. Ersoy, "Performance evaluation of wireless sensor networks in realistic wildfire simulation scenarios," in *Proceedings of the 16th ACM International Conference on Modeling, Analysis & Simulation of Wireless and Mobile Systems*, ser. MSWiM '13. New York, NY, USA: ACM, 2013, pp. 109–118. [Online]. Available: http://doi.acm.org/10.1145/2507924.2507941

[6] W.-Z. Song, R. Huang, M. Xu, B. Shirazi, and R. Lahusen, "Design and deployment of sensor network for real-time high-fidelity volcano monitoring," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 21, no. 11, pp. 1658–1674, Nov 2010.

[7] S. Nittel, N. Trigoni, K. Ferentinos, F. Neville, A. Nural, and N. Pettigrew, "A drift-tolerant model for data management in ocean sensor networks," in *Proceedings of the 6th ACM International Workshop on Data Engineering for Wireless and Mobile Access*, ser. MobiDE '07. New York, NY, USA: ACM, 2007, pp. 49–58. [Online]. Available: http://doi.acm.org/10.1145/1254850.1254860

[8] J. Al-Karaki and A. Kamal, "Routing techniques in wireless sensor networks: a survey," *IEEE Wireless Communications*, vol. 11, no. 6, pp. 6–28, Dec 2004.

[9] S. Souiki, M. Feham, M. Feham, and N. Labraoui, "Geographic routing protocols for underwater wireless sensor networks: a survey," *CoRR*, vol. abs/1403.3779, 2014. [Online]. Available: http://arxiv.org/abs/1403.3779

[10] A. M. Popescua, I. G. Tudorachea, B. Pengb, and A. Kempa, "Surveying position based routing protocols for wireless sensor and ad-hoc networks," *International Journal of Communication Networks and Information Security (IJCNIS)*, vol. 4, no. 1, pp. 41–67, 2004.

[11] R. Rajagopalan and P. Varshney, "Data-aggregation techniques in sensor networks: A survey," *IEEE Communications Surveys Tutorials*, vol. 8, no. 4, pp. 48–63, Fourth 2006.

[12] C. Chen, "Location-based data aggregation in mobile ad hoc networks," Master's thesis, Universität Stuttgart, Holzgartenstr. 16, 70174 Stuttgart, 2003. [Online]. Available: http://elib.uni-stuttgart.de/opus/volltexte/2004/1841

[13] J. Tian, P. Marrón, and K. Rothermel, "Location-based hierarchical data aggregation in vehicular ad hoc networks," in *Kommunikation in Verteilten Systemen (KiVS)*, ser. Informatik aktuell, P. Müller, R. Gotzhein, and J. Schmitt, Eds. Springer Berlin Heidelberg, 2005, pp. 166–177. [Online]. Available: http://dx.doi.org/10.1007/3-540-27301-8_14

[14] B. Hofmann-Wellenhof, H. Lichtenegger, and E. Wasle, *GNSS - Global Navigation Satellite Systems: GPS, GLONASS, Galileo, and more*. Springer Vienna, November 2007.

[15] S. Gleason and D. Gebre-Egziabher, *GNSS Applications and Methods (GNSS Technology and Applications)*. Artech House Publishers, August 2009.

[16] M. Erol-Kantarci, H. Mouftah, and S. Oktug, "A Survey of Architectures and Localization Techniques for Underwater Acoustic Sensor Networks," *IEEE Communications Surveys Tutorials*, vol. 13, no. 3, pp. 487–502, Third 2011.

[17] M. Bal, M. Liu, W. Shen, and H. Ghenniwa, "Localization in cooperative Wireless Sensor Networks: A review," in *13th International Conference on Computer Supported Cooperative Work in Design (CSCWD).*, April 2009, pp. 438–443.

[18] L. Cheng, C. Wu, Y. Zhang, H. Wu, M. Li, and C. Maple, "A Survey of Localization in Wireless Sensor Network," *International Journal of Distributed Sensor Networks*, vol. 2012, 2012.

[19] U. Nazir, M. Arshad, N. Shahid, and S. Raza, "Classification of localization algorithms for wireless sensor network: A survey," in *2012 International Conference on Open Source Systems and Technologies (ICOSST)*, Dec 2012, pp. 1–5.

[20] S. Gu, Y. Yue, C. Maple, C. Wu, and B. Liu, "Challenges in mobile localisation in wireless sensor networks for disaster scenarios," in *2013 19th International Conference on Automation and Computing (ICAC)*, Sept 2013, pp. 1–6.

[21] S. Jauregui and M. Siller, "A big picture on localization algorithms considering sensor logic location," in *IEEE International Conference on Systems, Man and Cybernetics, 2009. SMC 2009.*, Oct 2009, pp. 734–739.

[22] Y.-C. Kan, S.-Y. Chiang, and C.-J. Lin, "A GPS anchor node for outdoor wireless sensor network applications," in *IEEE International Symposium on Radio-Frequency Integration Technology, 2009. RFIT 2009.*, Jan 2009, pp. 40–43.

[23] W. P. B. Jr, "Simulation, development and deployment of mobile wireless sensor networks for migratory bird tracking," 2012.

[24] D. Anthony, W. Bennett, M. Vuran, M. Dwyer, S. Elbaum, A. Lacy, M. Engels, and W. Wehtje, "Sensing through the continent: Towards monitoring migratory birds using cellular sensor networks," in *Proceedings of the* 11^{th} *ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, April 2012, pp. 329–340.

[25] B. Li, M. Mutschlechner, R. Kapitza, and F. Dressler, "Improving Transmission Reliability in Sensor Networks for Energy-Constrained Wildlife Monitoring," in *24th ACM Symposium on Operating Systems Principles (SOSP), Diversity Workshop, Poster Session*. Farmington, PA: ACM, November 2013.

[26] F. Dressler, S. Ripperger, M. Hierold, T. Nowak, C. Eibel, B. Cassens, F. Mayer, K. Meyer-Wegener, and A. Koelpin, "From Radio Telemetry to Ultra-Low Power Sensor Networks - Tracking Bats in the Wild," *IEEE Communications Magazine*, 2015, to appear.

[27] F. Reichenbach, M. H, and D. Timmermann, "Monitoring the ocean environment with large-area wireless sensor networks," in *Proceedings of the 8th EUROMICRO Conference on Digital System Design (DSD 2005)*, 2005.

[28] H. Wang, Z. Gao, Y. Guo, and Y. Huang, "A survey of range-based localization algorithms for cognitive radio networks," in *2012 2nd International Conference on Consumer Electronics, Communications and Networks (CECNet)*, April 2012, pp. 844–847.

[29] R. Stoleru, T. He, J. A. Stankovic, and D. Luebke, "A high-accuracy, low-cost localization system for wireless sensor networks," in *Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems*, ser. SenSys '05. New York, NY, USA: ACM, 2005, pp. 13–26. [Online]. Available: http://doi.acm.org/10.1145/1098918.1098921

[30] R. Peng and M. L. Sichitiu, "Probabilistic Localization for Outdoor Wireless Sensor Networks," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 11, no. 1, pp. 53–64, Jan. 2007. [Online]. Available: http://doi.acm.org/10.1145/1234822.1234823

[31] R. Crepaldi, P. Casari, A. Zanella, and M. Zorzi, "Testbed Implementation and Refinement of a Range-based Localization Algorithm for Wireless Sensor Networks," in *Proceedings of the 3rd International Conference on Mobile Technology, Applications & Systems*, ser. Mobility '06. New York, NY, USA: ACM, 2006. [Online]. Available: http://doi.acm.org/10.1145/1292331.1292401

[32] M. Hinkfoth, E. Heinrich, S. Vorköper, V. Kühn, and R. Salomon, "X-ORCA: FPGA-based Wireless Localization in the Sub-millimeter Range," in *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, ser. FPGA '12. New York, NY, USA: ACM, 2012, pp. 29–32. [Online]. Available: http://doi.acm.org/10.1145/2145694.2145700

[33] O. Adewumi, K. Djouani, and A. Kurien, "RSSI Based Indoor and Outdoor Distance Estimation for Localization in WSN," in *IEEE International Conference on Industrial Technology (ICIT)*, Feb 2013, pp. 1534–1539.

[34] A. T. Parameswaran, M. I. Husain, and S. Upadhyaya, "Is RSSI a Reliable Parameter in Sensor Localization Algorithms - An Experimental Study," in *28th International Symposium on Reliable Distributed Systems*, 2009.

[35] N. A. Dieng, C. Chaudet, M. Charbit, L. Toutain, and T. Ben Meriem, "Experiments on the RSSI as a Range Estimator for Indoor Localization," in *5th International Conference on New Technologies, Mobility and Security (NTMS), 2012*, May 2012, pp. 1–5.

[36] L. Hu and D. Evans, "Localization for Mobile Sensor Networks," in 10^{th} *Annual International Conference on Mobile Computing and Networking (MobiCom 2004)*, Phildadelphia, USA, 2004, pp. 45–57.

[37] J. Horneber and A. Hergenroder, "A Survey on Testbeds and Experimentation Environments for Wireless Sensor Networks," *IEEE Communications Surveys Tutorials*, vol. 16, no. 4, pp. 1820–1838, Fourthquarter 2014.

[38] *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, IEEE Standards Association, March 2012.

[39] Lincoln Laboratory, MASSACHUSETTS INSTITUTE OF TECHNOLOGY, "Distributed Sensor Networks," Tech. Rep., 1986.

[40] J. M. Kahn, R. H. Katz, and K. S. J. Pister, "Next Century Challenges: Mobile Networking for &Ldquo;Smart Dust&Rdquo;," in *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking*, ser. MobiCom '99. New York, NY, USA: ACM, 1999, pp. 271–278.

[41] F. Bajaber, "Large Scale Environmental Monitoring and Maintaining Sensing Coverage in Sensor Networks," in *International Conference on Future Internet of Things and Cloud (FiCloud)*, Aug 2014, pp. 253–257.

[42] R. Mittal and M. Bhatia, "Wireless sensor networks for monitoring the environmental activities," in *IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)*, Dec 2010, pp. 1–5.

[43] V. Escorza and F. Guedea, "A Wireless Sensors Network Development for Environmental Monitoring Using OPC Unified Architecture in a Generic Manufacturing System," in *International Conference on Mechatronics, Electronics and Automotive Engineering (ICMEAE)*, Nov 2014, pp. 187–192.

[44] W. Jing and L. Tingting, "Application of wireless sensor network in yangtze river basin water environment monitoring," in *27th Chinese Control and Decision Conference (CCDC), 2015*, May 2015, pp. 5981–5985.

[45] M. Chayon, T. Rahman, M. Rabbi, and M. Masum, "Automated river monitoring system for bangladesh using wireless sensor network," in *10th International Conference on Computer and Information Technology, 2007. (ICCIT) 2007.*, Dec 2007, pp. 1–6.

[46] R. Herlien, T. O'Reilly, K. Headley, D. Edgington, S. Tilak, T. Fountain, and P. Shin, "An ocean observatory sensor network application," in *IEEE Sensors, 2010*, Nov 2010, pp. 1837–1842.

[47] M. Fazio, M. Paone, A. Puliafito, and M. Villari, "Heterogeneous sensors become homogeneous things in smart cities," in *6th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2012*, July 2012, pp. 775–780.

[48] B. Molina, C. Palau, G. Fortino, A. Guerrieri, and C. Savaglio, "Empowering smart cities through interoperable sensor network enablers," in *IEEE International Conference on Systems, Man and Cybernetics (SMC), 2014*, Oct 2014, pp. 7–12.

[49] F. Juraschek, A. Zubow, O. Hahm, M. Scheidgen, B. Blywis, R. Sombrutzki, M. Gunes, and J. Fischer, "Towards Smart Berlin - an experimental facility for heterogeneous Smart City infrastructures," in *IEEE 37th Conference on Local Computer Networks Workshops (LCN Workshops), 2012*, Oct 2012, pp. 886–892.

[50] T. Gea, J. Paradells, M. Lamarca, and D. Roldan, "Smart Cities as an Application of Internet of Things: Experiences and Lessons Learnt in Barcelona," in *Seventh International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2013*, July 2013, pp. 552–557.

[51] I. Benkhelifa, N. Nouali-Taboudjemat, and S. Moussaoui, "Disaster Management Projects Using Wireless Sensor Networks: An Overview," in *28th International Conference on Advanced Information Networking and Applications Workshops (WAINA), 2014*, May 2014, pp. 605–610.

[52] G. Ranjan and A. Kumar, "A natural disasters management system based on location aware distributed sensor networks," in *IEEE International Conference on Mobile Adhoc and Sensor Systems Conference, 2005.*, Nov 2005, pp. 3 pp.–182.

[53] S. Sana and M. Matsumoto, "A Wireless Sensor Network Protocol for Disaster Management," in *Information, Decision and Control, 2007. IDC '07*, Feb 2007, pp. 209–213.

[54] R. Singh and G. Asutkar, "Survey on various wireless sensor network techniques for monitoring activities of wild animals," in *International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS), 2015*, March 2015, pp. 1–5.

[55] A. de la Piedra, F. Benitez-Capistros, F. Dominguez, and A. Touhafi, "Wireless sensor networks for environmental research: A survey on limitations and challenges," in *IEEE EUROCON, 2013*, July 2013, pp. 267–274.

[56] K. Malarvizhi, M. Brindha, and M. Kumar, "Evaluation of energy efficient routing in wireless multimedia sensor networks," in *2nd International Conference on Electronics and Communication Systems (ICECS), 2015*, Feb 2015, pp. 1387–1391.

[57] Y. Kashio, S. Matsumoto, S. Tokunaga, S. Saiki, and M. Nakamura, "Design and implementation of service framework for presence sensing in home network system," in *Third International Conference on Digital Information, Networking, and Wireless Communications (DINWC), 2015*, Feb 2015, pp. 109–114.

[58] S. Nourizadeh, C. Deroussent, Y. Song, and J. Thomesse, "Medical and Home Automation Sensor Networks for Senior Citizens Telehomecare," in *IEEE International Conference on Communications Workshops, 2009. ICC Workshops 2009.*, June 2009, pp. 1–5.

[59] Y. Li, J. Maorong, G. Zhenru, Z. Weiping, and G. Tao, "Design of Home Automation System Based on ZigBee Wireless Sensor Network," in *1st International Conference on Information Science and Engineering (ICISE), 2009*, Dec 2009, pp. 2610–2613.

[60] S. H. Lee, S. Lee, H. Song, and H. S. Lee, "Wireless sensor network design for tactical military applications : Remote large-scale environments," in *Proceedings of the 2009 IEEE Military Communications Conference (MILCOM), 2009*, Oct 2009, pp. 1–7.

[61] M. Durisic, Z. Tafa, G. Dimic, and V. Milutinovic, "A Survey of Military Applications of Wireless Sensor Networks," in *Proceedings of the 2012 Mediterranean Conference on Embedded Computing (MECO)*, June 2012, pp. 196–199.

[62] J. Stankovic, A. Wood, and T. He, "Realistic Applications for Wireless Sensor Networks," in *Theoretical Aspects of Distributed Computing in Sensor Networks*, ser. Monographs in Theoretical Computer Science. An EATCS Series, S. Nikoletseas and J. D. Rolim, Eds. Springer Berlin Heidelberg, 2011, pp. 835–863.

[63] L. Inc., "Libelium Case Studies," 2015, http://www.libelium.com/libeliumworld/case-studies/.

[64] K. Yanrong, Z. Lei, C. Cai, G. Yuming, L. Hao, C. Ying, J. Whyte, and T. Hart, "Comparative Study of Smart Cities in Europe and China," DG CNECT, EU Commission with China Academy of Telecommunications Research (CATR), Tech. Rep., 2014.

[65] M. Ramesh, N. Vasudevan, , and J. Freeman, "Real Time Landslide Monitoring via Wireless Sensor Network," in *Proceedings of the EGU General Assembly 2009*, 2009.

[66] M. V. Ramesh, "Real-time Wireless Sensor Network for Landslide Detection," in *Proceedings of the 2009 3rd International Conference on Sensor Technologies and Applications*, July 2009.

[67] R. E. Bryant and D. R. O'Hallaron, *Computer Systems: A Programmer's Perspective*, 3rd ed. Pearson Education Limited, 2015.

[68] *Waspmote Datasheet, Document version: v5.4 - 06/2015*, Libelium Comunicaciones Distribuidas S.L, 2015.

[69] *MPR2400CB Datasheet, 6020-0060-04 Rev B*, Memsic, Inc., 2007.

[70] *XM2110CB Datasheet, 6020-0124-01 Rev B*, Memsic, Inc., 2007.

[71] "Advanticsys website," Advanticsys, June 2015. [Online]. Available: http://www.advanticsys.com/shop/mtmcm3000msp-p-6.html

[72] *TPR2420 Datasheet, 6020-0094-02 Rev B*, Memsic, Inc., 2007.

[73] *MSP430F15x, MSP430F16x, MSP430F161x MIXED SIGNAL MICROCONTROLLER Datasheet, SLAS368G*, Texas Instruments, 2011.

[74] *CC2420 Datasheet, SWRS041c*, Texas Instruments, 2013.

[75] *Fractus Micro Reach Xtend Bluetooth, Zigbee, 802.11b/g WLAN Chip Antenna*, FRACTUS, S.A., October 2007.

[76] J. Lu, H. Okada, T. Itoh, R. Maeda, and T. Harada, "Assembly of Super Compact Wireless Sensor Nodes for Environmental Monitoring Applications," in *Proceedings of the 2013 Symposium on Design, Test, Integration and Packaging of MEMS/MOEMS (DTIP)*, April 2013, pp. 1–4.

[77] R. R. Roy, *Handbook of Mobile Ad Hoc Networks for Mobility Models*, 1st ed. Springer US, 2001.

[78] T. Camp, J. Boleng, and V. Davies, "A survey of mobility models for ad hoc network research," *Wireless Communications & Mobile Computing (WCMC): Special Issue on Mobile Ad Hoc Networking: Research, Trends and Applications*, vol. 2, pp. 483–502, 2002.

[79] F. Bai, N. Sadagopan, and A. Helmy, "Important: a framework to systematically analyze the impact of mobility on performance of routing protocols for adhoc networks," in *INFOCOM 2003. 22nd Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, vol. 2, March 2003, pp. 825–835 vol.2.

[80] J. Tian, J. Haĺhner, C. Becker, I. Stepanov, and K. Rothermel, "Graph-based mobility model for mobile ad hoc network simulation," in *Proceedings of the 35th Annual Simulation Symposium, 2002.*, April 2002, pp. 337–344.

[81] P. Kohlstock, *Kartographie*, 3rd ed. UTB GmbH, 2014.

[82] NIMA, "World geodetic system 1984," National Imagery and Mapping Agency, East Lansing, Michigan, Tech. Rep. TR8350.2, January 2000.

[83] *NAVSTAR Global Positioning Service Survey*, US Army Corps of Engineers, July 2003.

[84] H. Forster, *Galileo : a positioning system : strategic, scientific and technical stakes*. Académie nationale de l'air et de l'espace, 2005.

[85] Jonathan Amos (BBC News), "Map illustrates 'Russian GPS' failure," April 2014. [Online]. Available: http://www.bbc.com/news/science-environment-26957569

[86] BBC News, "Galileo satellites go into wrong, lower orbit," August 2014. [Online]. Available: http://www.bbc.com/news/world-europe-28910662

[87] K. Betke, *The NMEA 0183 Protocol*, National Marine Electronics Association, August 2001.

[88] C. Wang and L. Xiao, "Sensor Localization under Limited Measurement Capabilities," *IEEE Network*, vol. 21, no. 3, pp. 16–23, May 2007.

[89] I. Guvenc and C.-C. Chong, "A Survey on TOA Based Wireless Localization and NLOS Mitigation Techniques," *IEEE Communications Surveys Tutorials*, vol. 11, no. 3, pp. 107–124, rd 2009.

[90] E. Engeler, *Foundations of Mathematics: Questions of Analysis, Geometry & Algorithmics*, 1st ed. Springer-Verlag, 1993.

[91] C. Levis, J. T. Johnson, and F. L. Teixeira, *Radiowave Propagation: Physics and Applications*, 1st ed. John Wiley & Sons, 2010.

[92] J. A. Shaw, "Radiometry and the friis transmission equation," *American Journal of Physics*, vol. 81, no. 1, pp. 33–37, 2013. [Online]. Available: http://scitation.aip.org/content/aapt/journal/ajp/81/1/10.1119/1.4755780

[93] R. P. Sawant, Q. Liang, D. O. Popa, and F. Lewis, "Experimental Path Loss Models for Wireless Sensor Networks," in *Proceedings of the 2007 IEEE Military Communications Conference (MILCOM)*, Oct 2007, pp. 1–7.

[94] G. Mao, B. Anderson, and B. Fidan, "Online Calibration of Path Loss Exponent in Wireless Sensor Networks," in *Proceedings of the 2006 IEEE Global Telecommunications Conference (GLOBECOM)*, Nov 2006, pp. 1–6.

[95] H.-C. Chen, T.-H. Lin, H. Kung, C.-K. Lin, and Y. Gwon, "Determining RF angle of arrival using COTS antenna arrays: A field evaluation," in *Proceedings of the 2012 IEEE Military Communications Conference (MILCOM)*, Oct 2012, pp. 1–6.

[96] *ATmega32/ATmega32L 8-bit Microcontroller with 32KBytes In-System Programmable Flash*, Atmel, 2011.

[97] Ibrahim M. M. El Emary and Dr. S. Ramakrishnan, *Wireless Sensor Networks: From Theory to Applications*, 1st ed. CRC Press, 2013.

[98] W. Du, D. Navarro, F. Mieyeville, and F. Gaffiot, "Towards a taxonomy of simulation tools for wireless sensor networks," in *Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques*, ser. SIMUTools '10. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2010, pp. 52:1–52:7. [Online]. Available: http://dx.doi.org/10.4108/ICST.SIMUTOOLS2010.8659

[99] C. Singh, O. Vyas, and M. Tiwari, "A Survey of Simulation in Sensor Networks," in *International Conference on Computational Intelligence for Modelling Control Automation, 2008*, Dec 2008, pp. 867–872.

[100] N. Binti Mohd Ngabas and J. Bin Abdullah, "A review of simulation framework for wireless sensor networks localization," in *2014 IEEE Student Conference on Research and Development (SCOReD)*, Dec 2014, pp. 1–6.

[101] M. Korkalainen, M. Sallinen, N. Karkkainen, and P. Tukeva, "Survey of wireless sensor networks simulation tools for demanding applications," in *2009 Fifth International Conference on Networking and Services (ICNS).*, April 2009, pp. 102–106.

[102] T. He, C. Huang, B. M. Blum, J. A. Stankovic, and T. Abdelzaher, "Range-free Localization Schemes for Large Scale Sensor Networks," in *Proceedings of the 9th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '03. New York, NY, USA: ACM, 2003, pp. 81–95.

[103] N. Bulusu, J. Heidemann, and D. Estrin, "GPS-less Low Cost Outdoor Localization For Very Small Devices," *IEEE Personal Communications*, vol. 7, no. 5, pp. 28–34, Oct 2000.

[104] A. S. Tannenbaum, *Computer Networks*, 4th ed. Prentice Hall, 2003.

[105] Scalable Network Techologies, "QualNet Product Page," October 2015. [Online]. Available: http://web.scalable-networks.com/content/qualnet

[106] K. Fall and K. Varadhan, *The ns Manuall*, The VINT project, November 2011.

[107] A. Varga, *OMNeT++ User manual*, OpenSim Ltd., 2014.

[108] D. Niculescu and B. Nath, "Ad hoc positioning system (aps) using aoa," in *INFOCOM 2003. 22nd Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, vol. 3, March 2003, pp. 1734–1743 vol.3.

[109] R. Nagpal, H. Shrobe, and J. Bachrach.

[110] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte carlo localization for mobile robots," in *Proceedings of the 1999 IEEE International Conference on Robotics and Automation*, vol. 2, 1999, pp. 1322–1328 vol.2.

[111] D. P. Kroese, T. Brereton, T. Taimre, and Z. I. Botev, "Why the monte carlo method is so important today," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 6, no. 6, pp. 386–392, 2014. [Online]. Available: http://dx.doi.org/10.1002/wics.1314

[112] A. Baggio and K. Langendoen, "Monte-carlo localization for mobile wireless sensor networks," in *Mobile Ad-hoc and Sensor Networks*, ser. Lecture Notes in Computer Science, J. Cao, I. Stojmenovic, X. Jia, and S. Das, Eds. Springer Berlin Heidelberg, 2006, vol. 4325, pp. 317–328.

[113] J. Yi, S. Yang, and H. Cha, "Multi-hop-based monte carlo localization for mobile sensor networks," in *4th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON).*, June 2007, pp. 162–171.

[114] M. Rudafshani and S. Datta, "Localization in Wireless Sensor Networks," in *6th International Symposium on Information Processing in Sensor Networks (IPSN), 2007*, April 2007, pp. 51–60.

[115] S. Zhang, J. Cao, C. Li-Jun, and D. Chen, "Accurate and Energy-Efficient Range-Free Localization for Mobile Sensor Networks," *IEEE Transactions on Mobile Computing*, vol. 9, no. 6, pp. 897–910, June 2010.

[116] G. Teng, K. Zheng, and W. Dong, "MA-MCL: Mobile-Assisted Monte Carlo Localization for Wireless Sensor Networks." *IJDSN*, vol. 2011, 2011.

[117] R. Huang and G. Záruba, "Monte carlo localization of wireless sensor networks with a single mobile beacon," *Wireless Networks*, vol. 15, no. 8, pp. 978–990, 2009.

[118] M. Mwila, K. Djouani, and A. Kurien, "An Efficient Approach to Node Localisation and Tracking in Wireless Sensor Networks," in *2014 IEEE Global Communications Conference (GLOBECOM)*, Dec 2014, pp. 492–497.

[119] H. Mistry and N. Mistry, "RSSI Based Localization Scheme in Wireless Sensor Networks: A Survey," in *2015 Fifth International Conference on Advanced Computing Communication Technologies (ACCT)*, Feb 2015, pp. 647–652.

[120] L. Li, Y. Wu, Y. Ren, and N. Yu, "A RSSI Localization Algorithm Based on Interval Analysis for Indoor Wireless Sensor Networks," in *IEEE International Conference on Green Computing and Communications and IEEE International Conference on Cyber, Physical and Social Computing (iThings/CPSCom)*, Aug 2013, pp. 434–437.

[121] K.-C. Wang, L. Tang, and Y. Huang, "Wireless Sensors on Rotating Structures: Performance Evaluation and Radio Link Characterization," in *Proceedings of the Second ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization*, ser. WinTECH '07. New York, NY, USA: ACM, 2007, pp. 3–10. [Online]. Available: http://doi.acm.org/10.1145/1287767.1287770

[122] C. Phillips, D. Sicker, and D. Grunwald, "A Survey of Wireless Path Loss Prediction and Coverage Mapping Methods," *IEEE Communications Surveys Tutorials*, vol. 15, no. 1, pp. 255–270, First 2013.

[123] Y. Chen and A. Terzis, "Calibrating RSSI Measurements for 802.15.4 Radios," in *Proceedings of the Seventh European Conference on Wireless Sensor Networks (EWSN)*, 2010.

[124] K. Benkic, M. Malajner, P. Planinsic, and Z. Cucej, "Using RSSI value for distance estimation in wireless sensor networks based on ZigBee," in *15th International Conference on Systems, Signals and Image Processing (IWSSIP) 2008*, June 2008, pp. 303–306.

[125] K. Srinivasan and P. Levis, "RSSI is Under Appreciated," in *In Proceedings of the Third Workshop on Embedded Networked Sensors (EmNets*, 2006.

[126] R.-H. Wu, Y.-H. Lee, H.-W. Tseng, Y.-G. Jan, and M.-H. Chuang, "Study of characteristics of RSSI signal," in *IEEE International Conference on Industrial Technology (ICIT) 2008*, April 2008, pp. 1–3.

[127] *CC2430 Datasheet, SWRS036F*, Texas Instruments, 2007.

[128] *CC2520 Datasheet, SWRS068*, Texas Instruments, 2007.

[129] *AVR2001: AT86RF230 Software Programmer's Guide Rev. 8087A-AVR-07/07*, Atmel, 2007.

[130] *TDA5250 D2 ASK/FSK 868MHz Wireless Tranceiver Data Sheet v1.7*, Infineon Technologies, 2007.

[131] *ETRX2 and ETRX2HR ZIGBEE Modules Product Manual v1.09*, TELEGESIS, 2009.

[132] *SX1211 Transceiver Ultra-Low Power Integrated UHF Transceiver Rev. 8*, Texas Instruments, 2013.

[133] S. FANG and Y. Yang, "The Impact of Weather Condition on Radio-based Distance Estimation: A Case Study in GSM Networks with Mobile Measurements," *IEEE Transactions on Vehicular Technology*, vol. PP, no. 99, pp. 1–1, 2015.

[134] J. Rak, "Design of weather disruption-tolerant wireless mesh networks," in *2012 International Telecommunications Network Strategy and Planning Symposium (NETWORKS)*, Oct 2012, pp. 1–6.

[135] K. Harb, A. Srinivasan, C. Huang, and B. Cheng, "Prediction method to maintain QoS in weather impacted wireless and satellite networks," in *IEEE International Conference on Systems, Man and Cybernetics, 2007. ISIC 2007*, Oct 2007, pp. 4008–4013.

[136] Y. Zhang and D. G. Michelson, "Impact of wind-induced fading on the capacity of point-to-multipoint fixed wireless access systems," in *Proceedings of the 2006 International Conference on Wireless Communications and Mobile Computing*, ser. IWCMC '06. New York, NY, USA: ACM, 2006, pp. 979–984. [Online]. Available: http://doi.acm.org/10.1145/1143549.1143745

[137] G. Zhou, T. He, S. Krishnamurthy, and J. A. Stankovic, "Impact of Radio Irregularity on Wireless Sensor Networks," in *Proceedings of the 2Nd International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '04. New York, NY, USA: ACM, 2004, pp. 125–138. [Online]. Available: http://doi.acm.org/10.1145/990064.990081

[138] M. Boban, T. Vinhoza, M. Ferreira, J. Barros, and O. Tonguz, "Impact of vehicles as obstacles in vehicular ad hoc networks," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 1, pp. 15–28, January 2011.

[139] TinyOS, "Tinyos website," http://www.tinyos.net/, accessed: 2015-07-30.

[140] P. Levis, *TinyOS Programming*, 1st ed. Cambridge University Press, 2011.

[141] Advanticsys, "Advanticsys website," http://www.advanticsys.com/, accessed: 2015-07-30.

[142] K. Seifert and O. Camacho, *Application Note AN3397*, Freescale Semiconductor, February 2007.

[143] *AN-1007 Estimating Velocity and Position Using Accelerometers*, CH Robotics, October 2012.

[144] *MPU-9150 Product Specification Revision 4.3*, InvenSense Inc., September 2013.

[145] *3-Axis Digital Compass IC HMC5883L*, Honeywell International Inc., February 2013.

[146] *Xtrinsic MAG3110 Three-Axis, Digital Magnetometer*, Freescale Semiconductor, Inc., February 2013.

[147] University of Virginia, "MCL Simulator Download," October 2015. [Online]. Available: http://www.cs.virginia.edu/mcl/download.html

[148] A. B. Bondi, "Characteristics of Scalability and Their Impact on Performance," in *Proceedings of the 2nd International Workshop on Software and Performance (WOSP)*, ser. WOSP '00. New York, NY, USA: ACM, 2000, pp. 195–203.

[149] *M9803R Bench Multimeter User Manual*, MASTECH, 2014.

[150] *Atmel ATmega640/V-1280/V-1281/V-2560/V-2561/V Datasheet*, Atmel, 2014.

[151] P. Crescenzi, M. D. Ianni, A. Marino, D. Merlini, G. Rossi, and P. Vocca, "Smooth movement and manhattan path based random waypoint mobility," *Information Processing Letters*, vol. 111, no. 5, pp. 239 – 246, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0020019010004011

[152] J. Kammann, M. Angermann, and B. Lami, "A new mobility model based on maps," in *2003 IEEE 58th Vehicular Technology Conference, 2003. VTC 2003-Fall.*, vol. 5, Oct 2003, pp. 3045–3049 Vol.5.

[153] C. Cheng, R. Jain, and E. van den Berg, "Wireless internet handbook," B. Furht and M. Ilyas, Eds. Boca Raton, FL, USA: CRC Press, Inc., 2003, ch. Location Prediction Algorithms for Mobile Wireless Systems, pp. 245–263. [Online]. Available: http://dl.acm.org/citation.cfm?id=989684.989696

[154] H. Sato, C. W. Berry, Y. Peeri, E. Baghoomian, G. L. Brendan E. Casey, J. M. VandenBrooks, J. F. Harrison, and M. M. Maharbiz, "Remote radio control of insect flight," *Frontiers in Integrative Neuroscience*, vol. 5, 2009.

[155] *ESRI Shapefile Technical Description*, Environmental Systems Research Institute, July 1998.

[156] M. Jain and H. Kandwal, "A Survey on Complex Wormhole Attack in Wireless Ad Hoc Networks," in *Proceedings of the 2009 International Conference on Advances in Computing, Control, and Telecommunication Technologies (ACT)*, ser. ACT '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 555–558.

[157] G. Luo, Z. Han, L. Lu, and M. Hussain, "Real-time and Passive Wormhole Detection for Wireless Sensor Networks," in *Proceedings of the 20th IEEE International Conference on Parallel and Distributed Systems (ICPADS)*, Dec 2014, pp. 592–599.

[158] T. Giannetsos, T. Dimitriou, and N. Prasad, "State of the Art on Defenses against Wormhole Attacks in Wireless Sensor Networks," in *1st International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace Electronic Systems Technology (Wireless VITAE)*, May 2009, pp. 313–318.

[159] Y. Zhang, W. Liu, W. Lou, and Y. Fang, "Location-Based Compromise-Tolerant Security Mechanisms for Wireless Sensor Networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 2, pp. 247–260, Feb 2006.

Appendix A

Supplemental Results

In the appendix additional results from the evaluation of both SA-MCL and PO-MCL are presented. They do not necessarily provide new findings, but prove that both protocols have evaluated in detail with lots of different parameter settings.

A.1 Additional Simulation Results for SA-MCL

A.1.1 Effect of Radio Range

Figures A.1-A.5 show the effect of an increasing radio range for different amounts of seed nodes. As explained in 5.4.2, an increasing seed density results in a smaller localization error in general. In addition to that, it can be found that increasing the radio range also has a strong effect on the localization error. It is strongly decreased up to radio ranges of 50 m. After that, only little improvement can be noted.

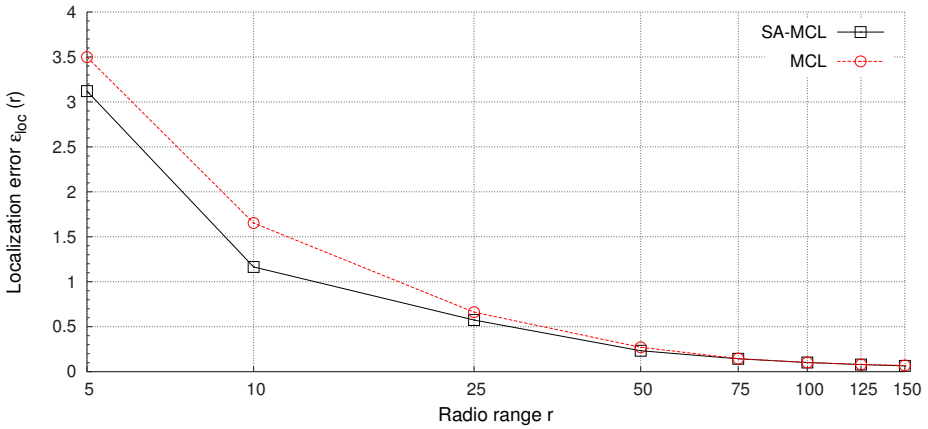


Figure A.1: Localization error ϵ_{loc} for $v_{\text{max}} = 20$, $N_{\text{sample}} = 25$, $N_{\text{nodes}} = 300$, $\rho_{\text{seed}} = 4.0$

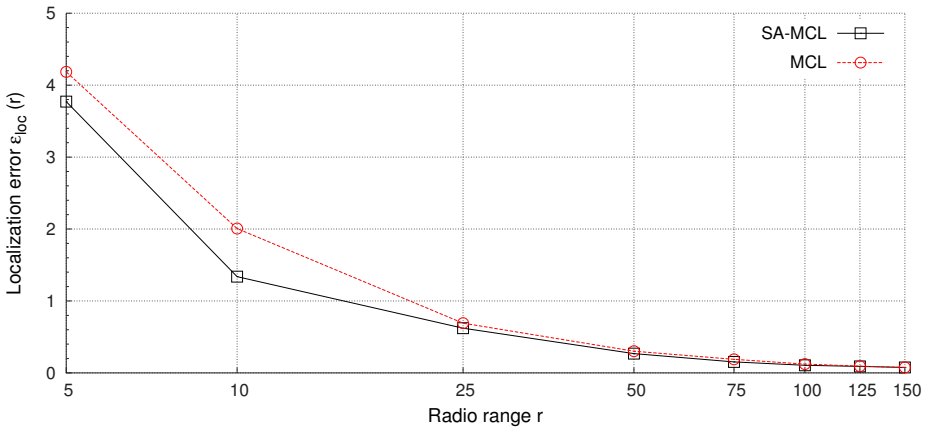


Figure A.2: Localization error ϵ_{loc} for $v_{\text{max}} = 20$, $N_{\text{sample}} = 25$, $N_{\text{nodes}} = 300$, $\rho_{\text{seed}} = 3.2$

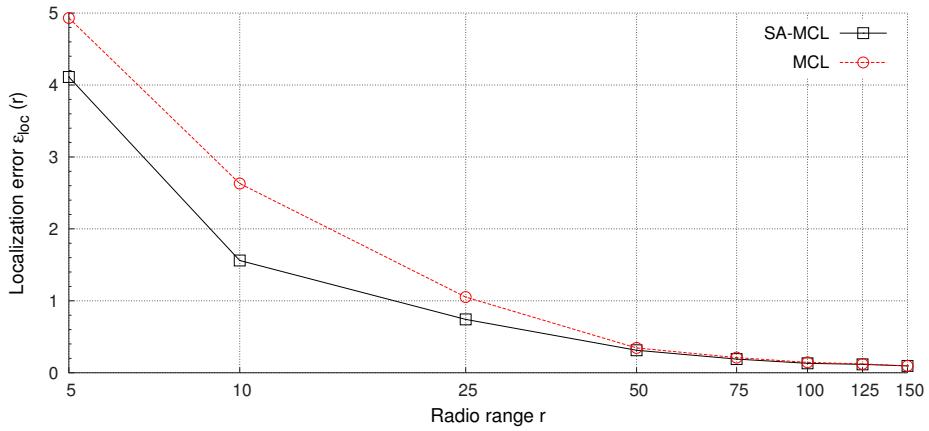


Figure A.3: Localization error ϵ_{loc} for $v_{\max} = 20$, $N_{\text{sample}} = 25$, $N_{\text{nodes}} = 300$, $\rho_{\text{seed}} = 2.4$

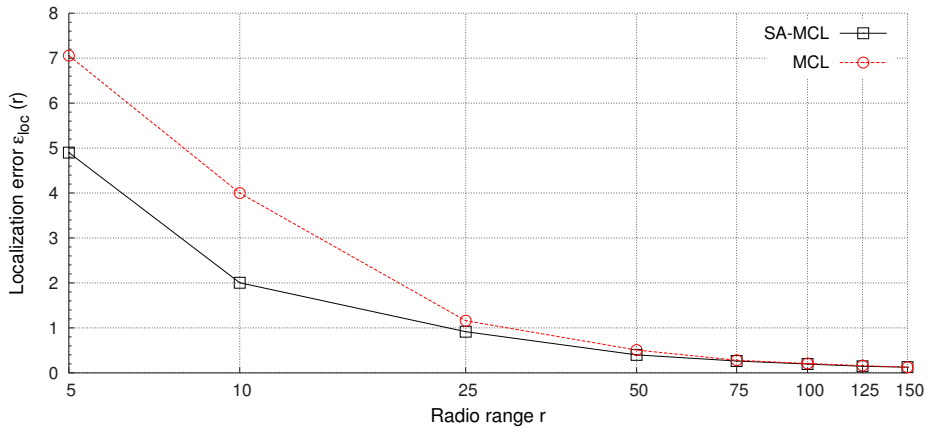


Figure A.4: Localization error ϵ_{loc} for $v_{\max} = 20$, $N_{\text{sample}} = 25$, $N_{\text{nodes}} = 300$, $\rho_{\text{seed}} = 1.6$

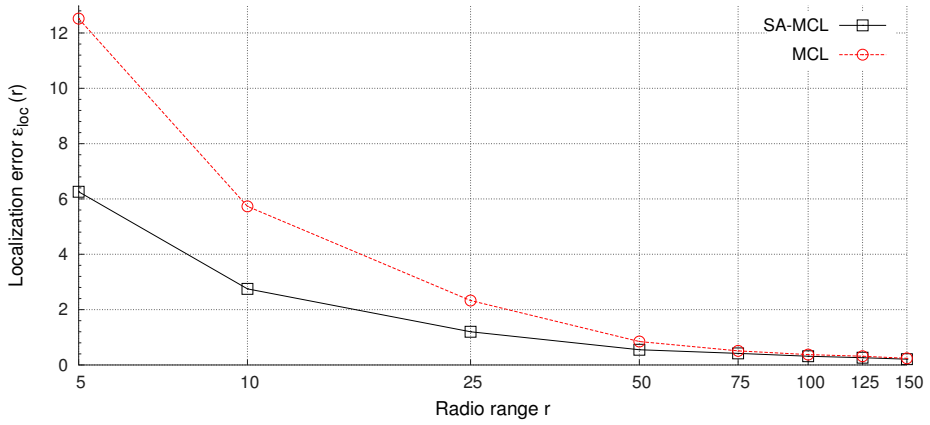


Figure A.5: Localization error ϵ_{loc} for $v_{\max} = 20$, $N_{\text{sample}} = 25$, $N_{\text{nodes}} = 300$, $\rho_{\text{seed}} = 0.8$

A.1.2 Scalability

Further results of the scalability analysis of SA-MCL can be found in Figures A.1 to A.5. As previously explained, an increasing number of nodes leads to small improvements of the localization error due to the more often forwarded localization announcements. The general characteristics of the curves in all figures stay the same, only the localization error is higher due to the continuously decreased seed density.

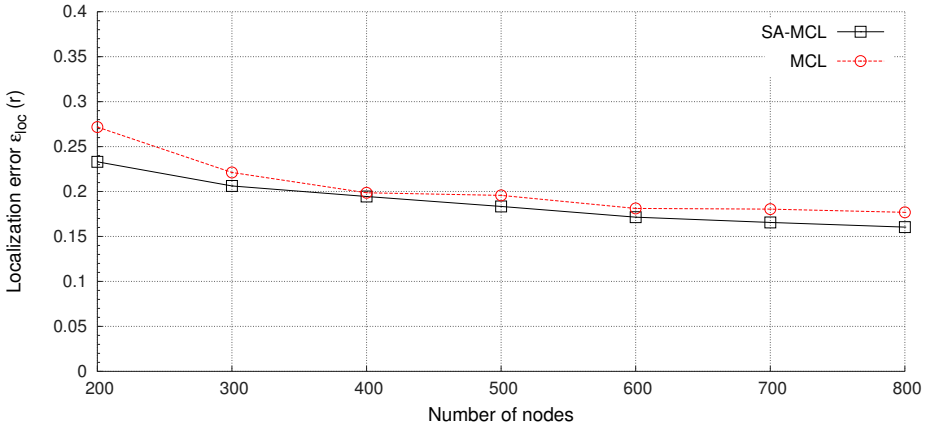


Figure A.1: Localization error ϵ_{loc} for $v_{\text{max}} = 20$, $r = 50$, $N_{\text{sample}} = 25$, $\rho_{\text{seed}} = 4.0$

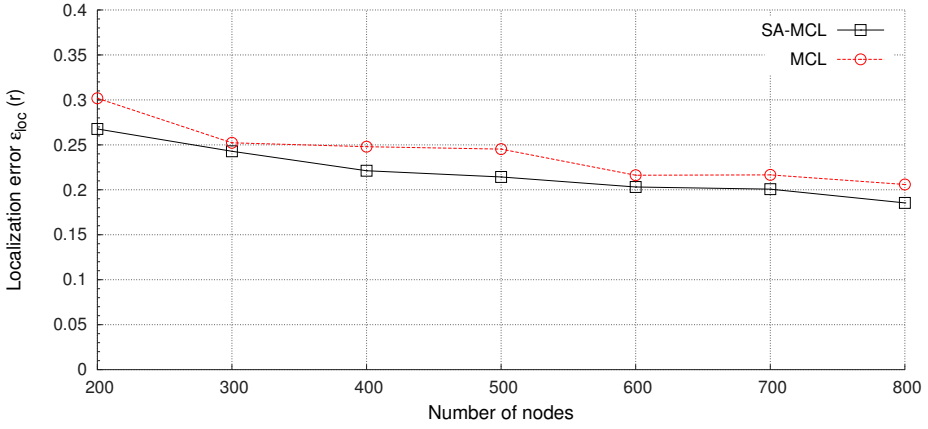


Figure A.2: Localization error ϵ_{loc} for $v_{\text{max}} = 20$, $r = 50$, $N_{\text{sample}} = 25$, $\rho_{\text{seed}} = 3.2$

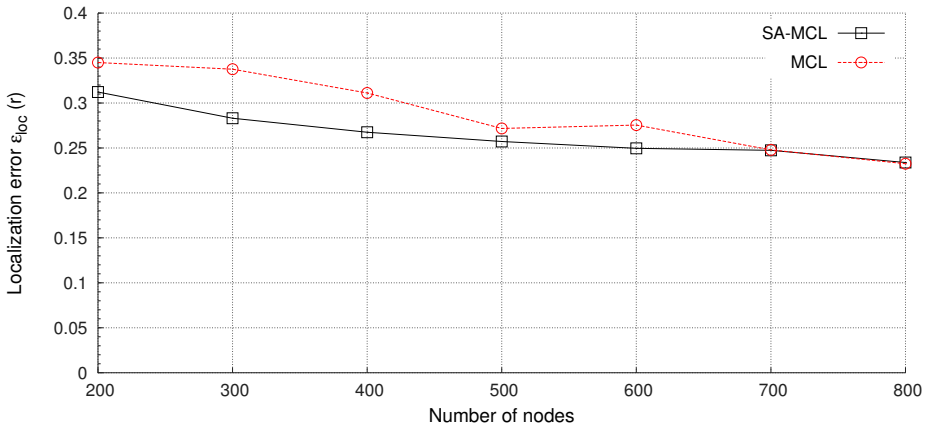


Figure A.3: Localization error ϵ_{loc} for $v_{\max} = 20$, $r = 50$, $N_{\text{sample}} = 25$, $\rho_{\text{seed}} = 2.4$

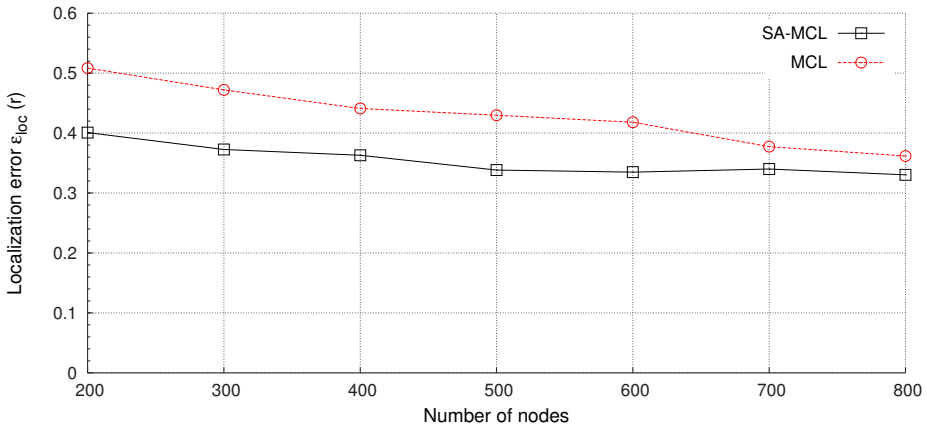


Figure A.4: Localization error ϵ_{loc} for $v_{\max} = 20$, $r = 50$, $N_{\text{sample}} = 25$, $\rho_{\text{seed}} = 1.6$

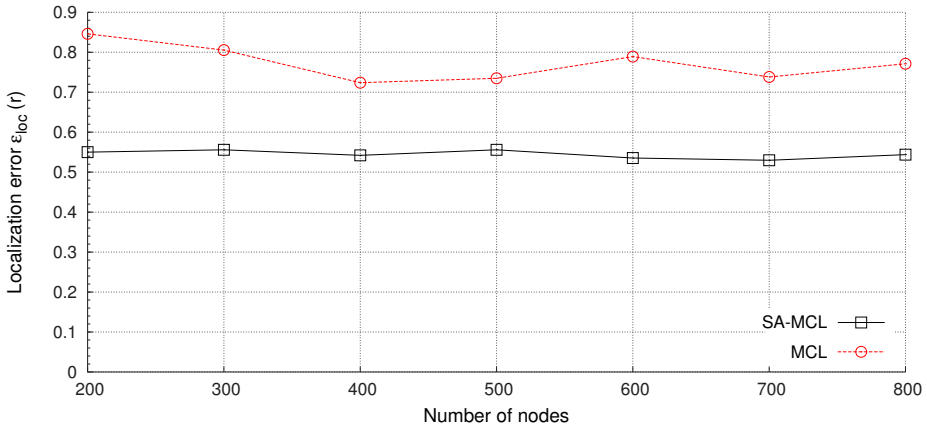


Figure A.5: Localization error ϵ_{loc} for $v_{\max} = 20$, $r = 50$, $N_{\text{sample}} = 25$, $\rho_{\text{seed}} = 0.8$

A.1.3 Effect of Node Velocity

Further results for varying node velocities are illustrated in Figures A.6 to A.11. In the beginning all curves show the characteristic behavior. Increasing velocity is benefiting the localization error until a certain threshold value. After that, the localization error increases due to more connection losses.

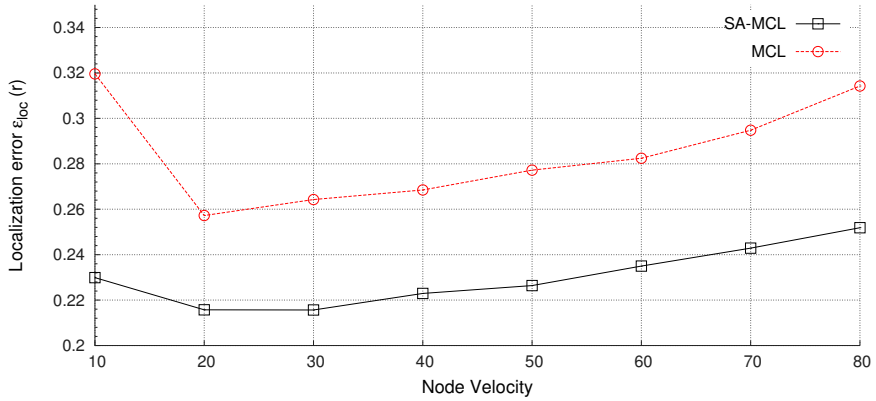


Figure A.6: Localization error ϵ_{loc} for $r = 50$, $N_{\text{sample}} = 25$, $\rho_{\text{seed}} = 4.0$, $N_{\text{nodes}} = 300$

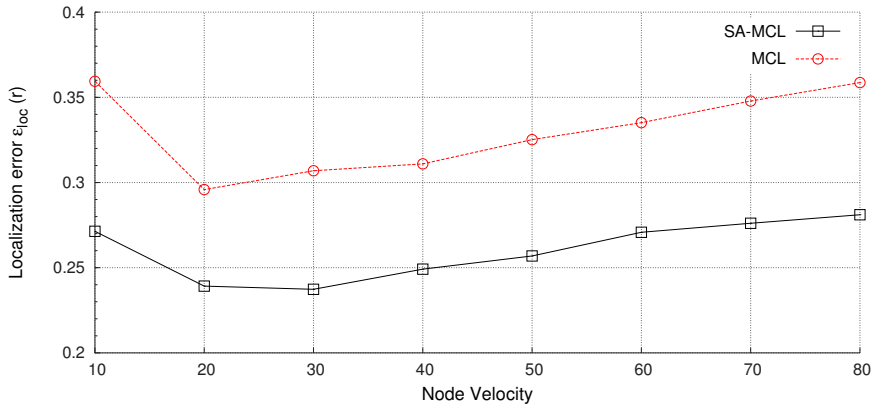


Figure A.7: Localization error ϵ_{loc} for $r = 50$, $N_{\text{sample}} = 25$, $\rho_{\text{seed}} = 3.2$, $N_{\text{nodes}} = 300$

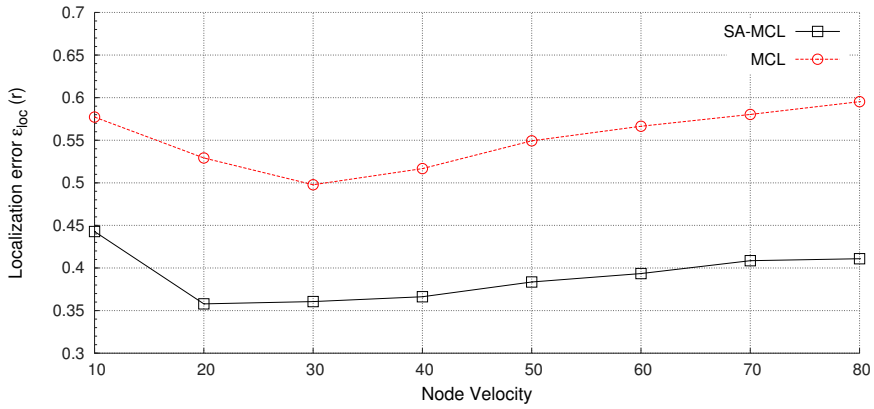


Figure A.8: Localization error ϵ_{loc} for $r = 50$, $N_{\text{sample}} = 25$, $\rho_{\text{seed}} = 1.6$, $N_{\text{nodes}} = 300$

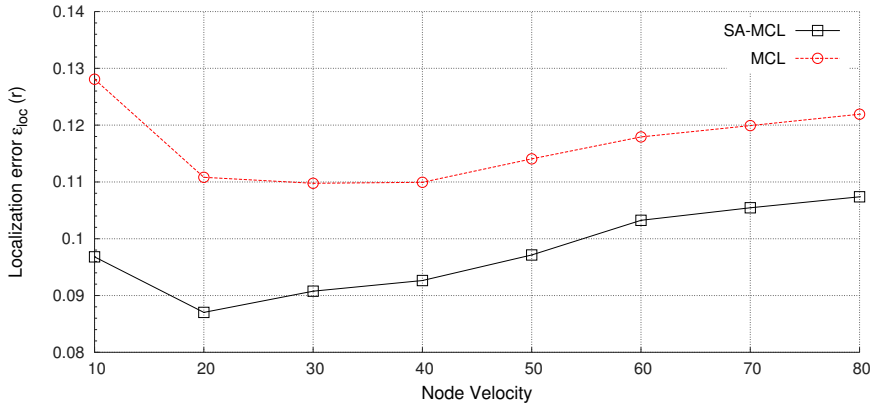


Figure A.9: Localization error ϵ_{loc} for $r = 100$, $N_{\text{sample}} = 25$, $\rho_{\text{seed}} = 4.0$, $N_{\text{nodes}} = 300$

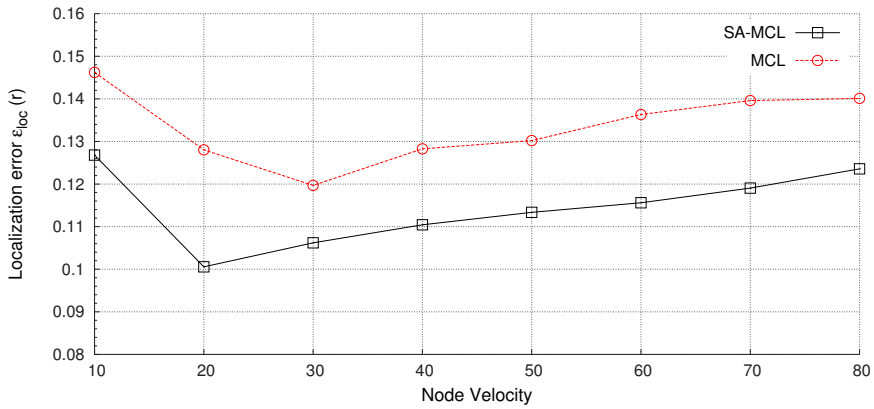


Figure A.10: Localization error ϵ_{loc} for $r = 100$, $N_{\text{sample}} = 25$, $\rho_{\text{seed}} = 3.2$, $N_{\text{nodes}} = 300$

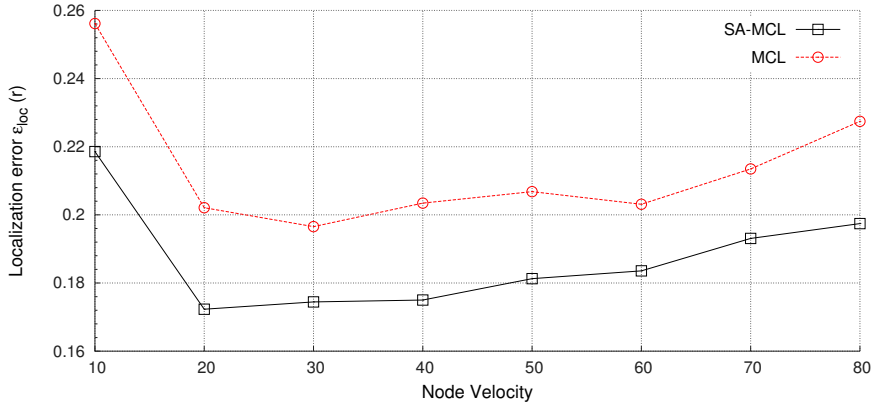
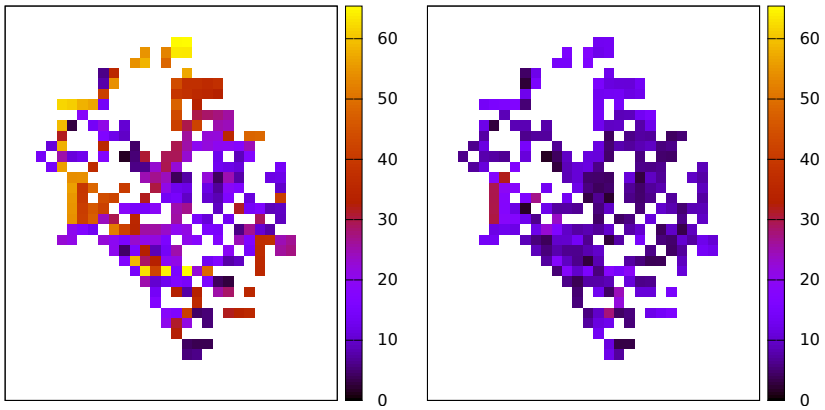


Figure A.11: Localization error ϵ_{loc} for $r = 100$, $N_{\text{sample}} = 25$, $\rho_{\text{seed}} = 1.6$, $N_{\text{nodes}} = 300$

A.2 Additional Field Test Results for SA-MCL

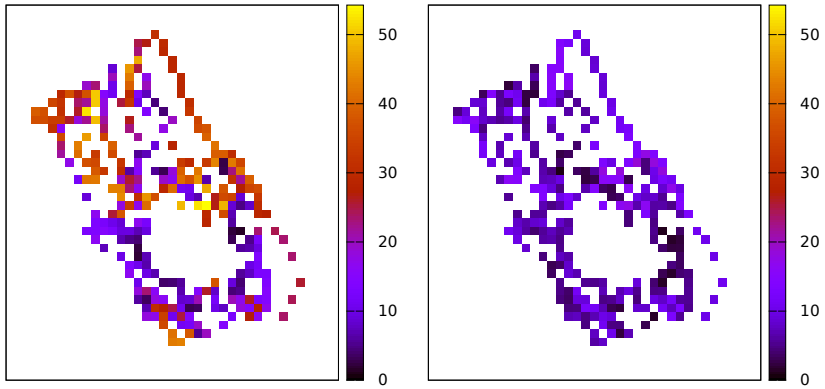
A.2.1 Grid Error Plots

The grid error plots of all experimental cars are shown in this section. In general, the performance of SA-MCL is much better compared to MCL. SA-MCL particularly provides better results at the outer regions of the test field.



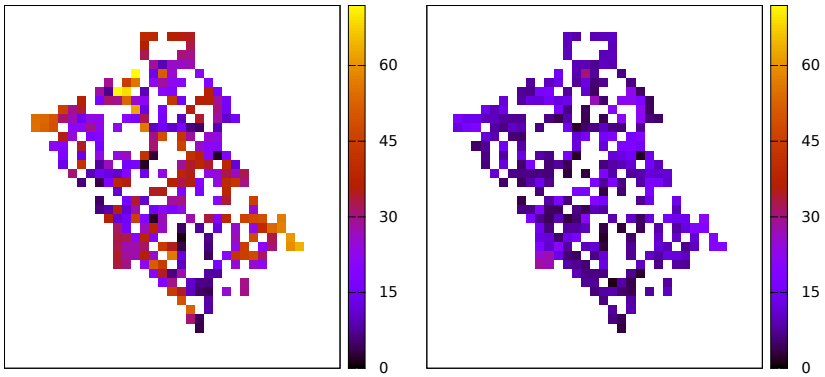
(a) MCL (b) SA-MCL

Figure A.12: Grid error for Car 1.



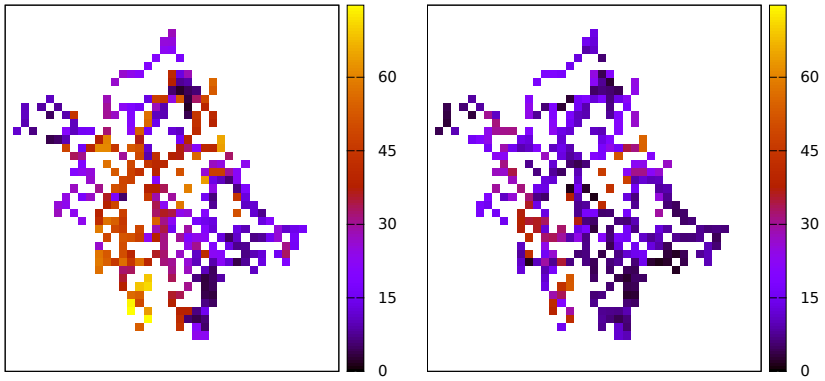
(a) MCL (b) SA-MCL

Figure A.13: Grid error for Car 2.



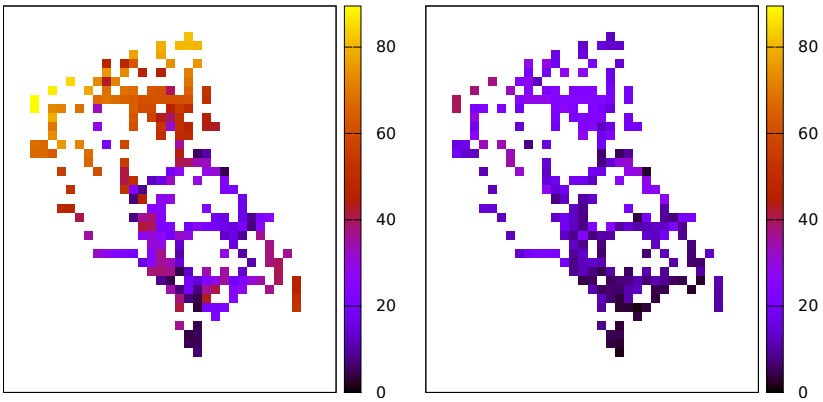
(a) MCL (b) SA-MCL

Figure A.14: Grid error for Car 3.



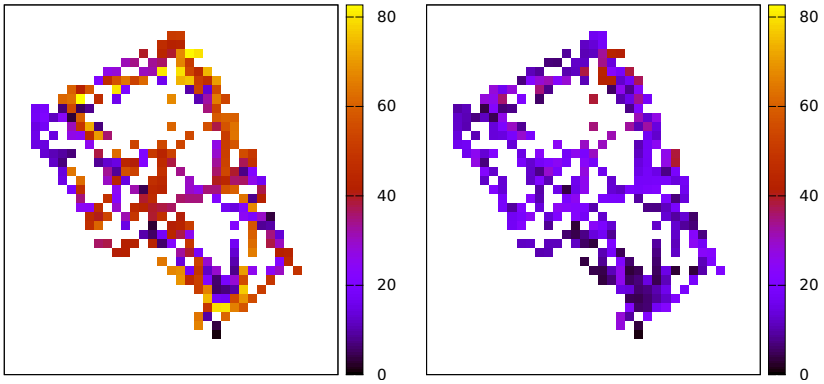
(a) MCL (b) SA-MCL

Figure A.15: Grid error for Car 4.



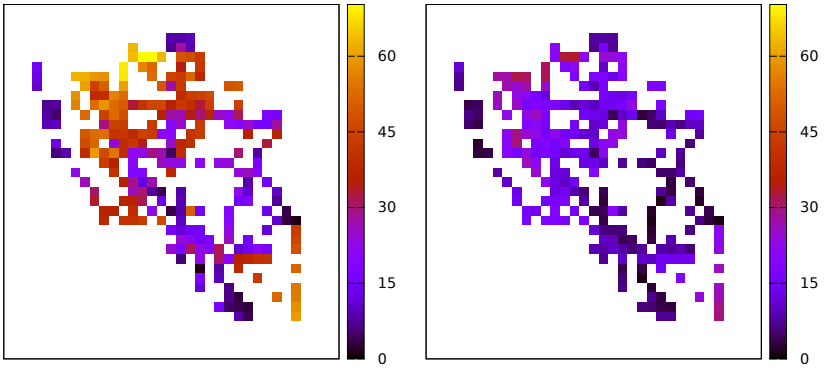
(a) MCL (b) SA-MCL

Figure A.16: Grid error for Car 5.



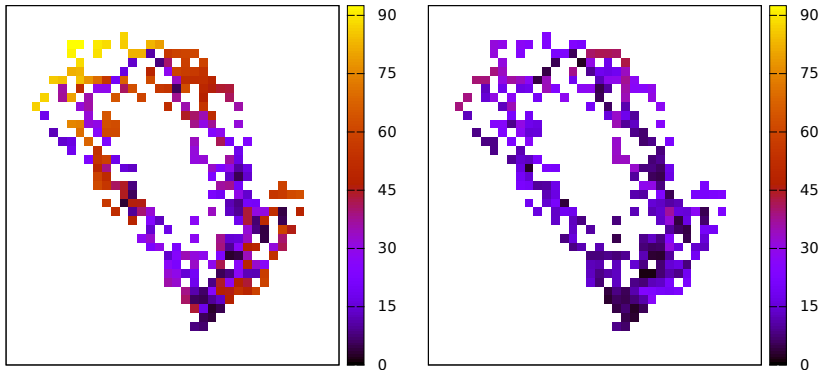
(a) MCL (b) SA-MCL

Figure A.17: Grid error for Car 6.



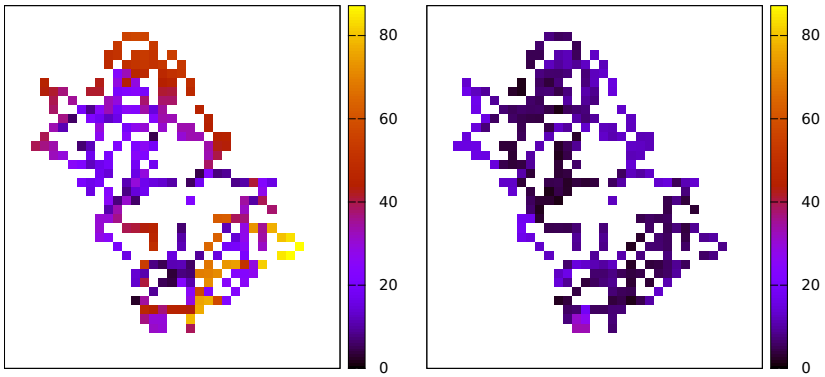
(a) MCL (b) SA-MCL

Figure A.18: Grid error for Car 7.



(a) MCL (b) SA-MCL

Figure A.19: Grid error for Car 8.

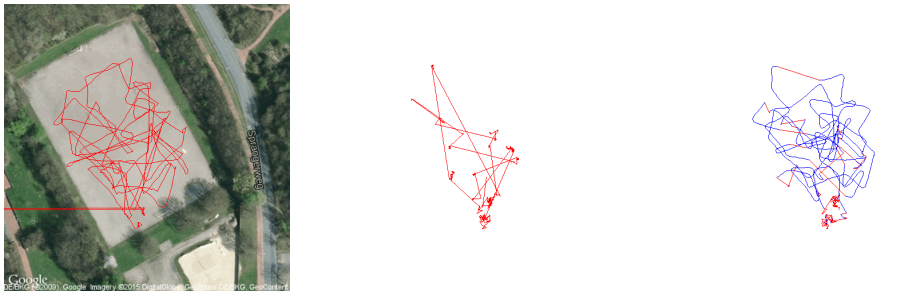


(a) MCL (b) SA-MCL

Figure A.20: Grid error for Car 9.

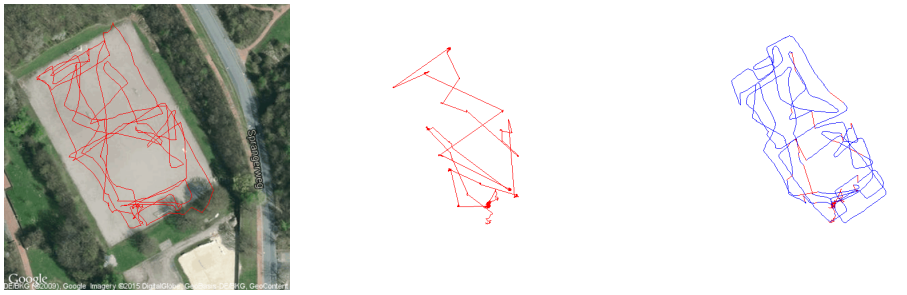
A.2.2 Path Traces

The path traces of all experimental cars are shown in this section. For SA-MCL blue color indicates the usage of SA-MCL and red color the usage of MCL. In general the path approximated by SA-MCL features many more similarities to the provided GPS ground truth data than MCL does. Consequently, all SA-MCL plots indicate that SA-MCL has been used more often than MCL.



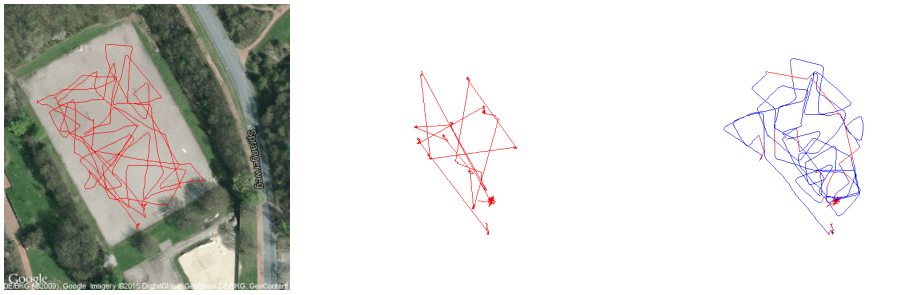
(a) GPS (b) MCL (c) SA-MCL

Figure A.21: Traces of GPS, MCL and SA-MCL for Car 1



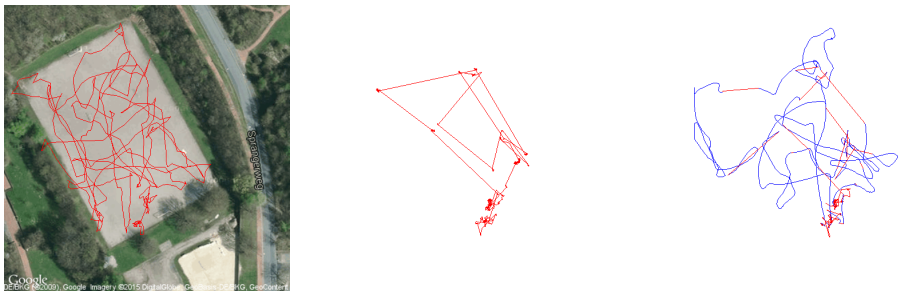
(a) GPS (b) MCL (c) SA-MCL

Figure A.22: Traces of GPS, MCL and SA-MCL for Car 2



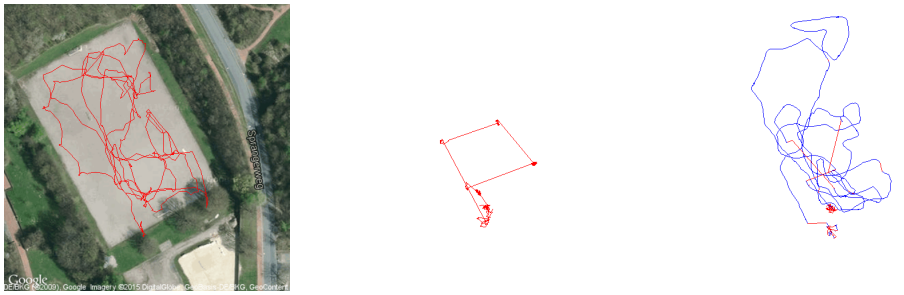
(a) GPS (b) MCL (c) SA-MCL

Figure A.23: Traces of GPS, MCL and SA-MCL for Car 3



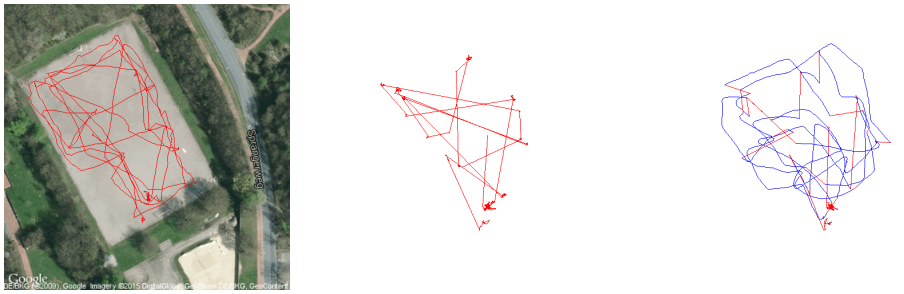
(a) GPS (b) MCL (c) SA-MCL

Figure A.24: Traces of GPS, MCL and SA-MCL for Car 4



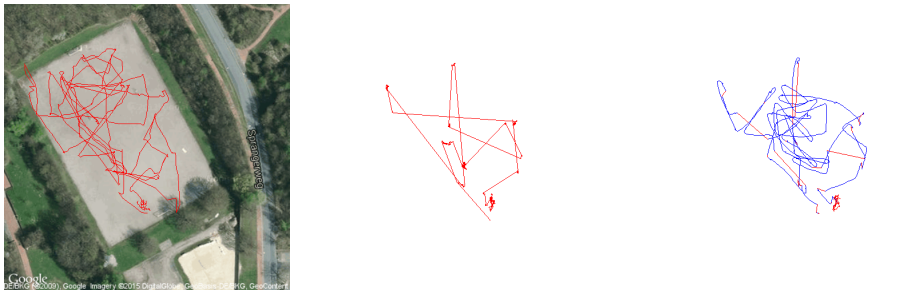
(a) GPS (b) MCL (c) SA-MCL

Figure A.25: Traces of GPS, MCL and SA-MCL for Car 5



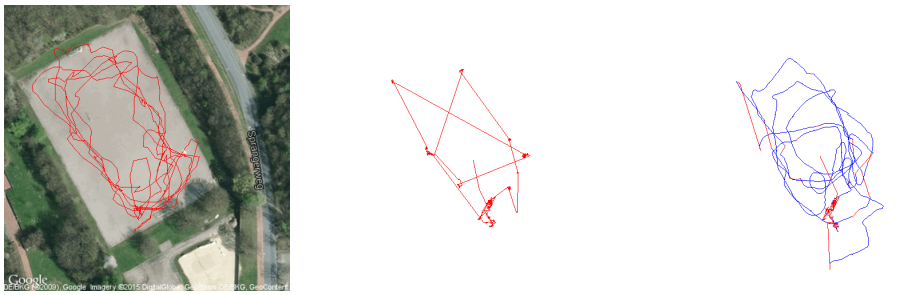
(a) GPS (b) MCL (c) SA-MCL

Figure A.26: Traces of GPS, MCL and SA-MCL for Car 6



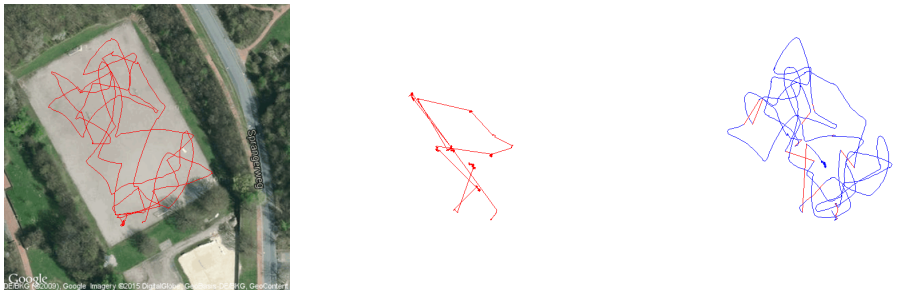
(a) GPS (b) MCL (c) SA-MCL

Figure A.27: Traces of GPS, MCL and SA-MCL for Car 7



(a) GPS (b) MCL (c) SA-MCL

Figure A.28: Traces of GPS, MCL and SA-MCL for Car 8



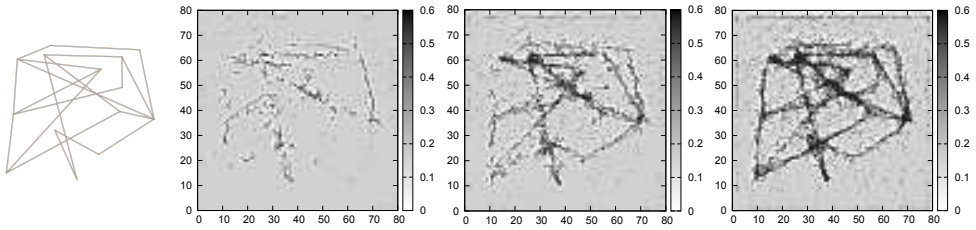
(a) GPS (b) MCL (c) SA-MCL

Figure A.29: Traces of GPS, MCL and SA-MCL for Car 9

A.3 Additional Results for PO-MCL

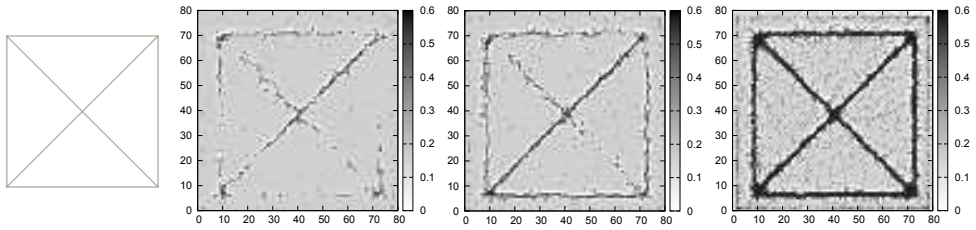
A.3.1 Path Convergence Graphs

Figures A.30- A.32 show the convergence process for additional path scenarios, which have been used to evaluate PO-MCL. In all scenarios the traveled paths are properly represented by the grid after 1440 min of simulation time. Of course the approximated paths of the square scenario shown in Figure A.31 are the plainest, as seed nodes and ordinary nodes are traveling mostly on the same paths and therefore the grid update technique is used more frequently.



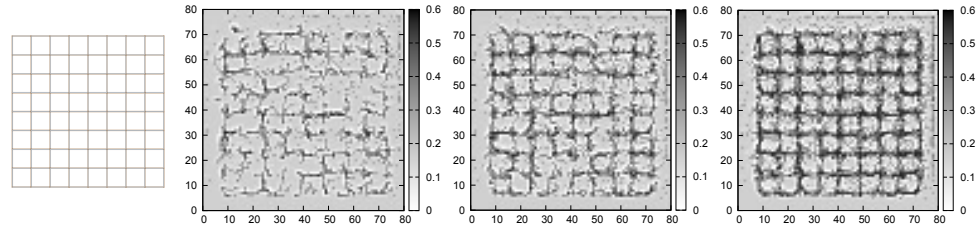
(a) Ground truth (b) After 360 min (c) After 720 min (d) After 1440 min

Figure A.30: Grid convergence for random path scenario.



(a) Ground truth (b) After 360 min (c) After 720 min (d) After 1440 min

Figure A.31: Grid convergence for square scenario



(a) Ground truth (b) After 360 min (c) After 720 min (d) After 1440 min

Figure A.32: Grid convergence for a grid scenario with 100 m^2 cell size

A.3.2 Simulation Results

A.3.2.1 Effect of Radio Range

Figures A.6- A.10 show the effect of the radio range for different amounts of seed nodes. As explained in Section 6.4.2, increasing the number of seed nodes results in a lower localization error in general. However, for larger radio ranges the localization error is growing, too. This is due to the implementation of PO-MCL and the simulation setup, which both do not account for congestion. In fact, the seed nodes will send their location announcements exactly at the same time which results in many packet collisions. The effect is amplified by the radio range of the nodes. The problem can be easily solved by applying a jittering mechanism to the process, i.e. each packet will be sent after a short random delay to avoid packet collisions.

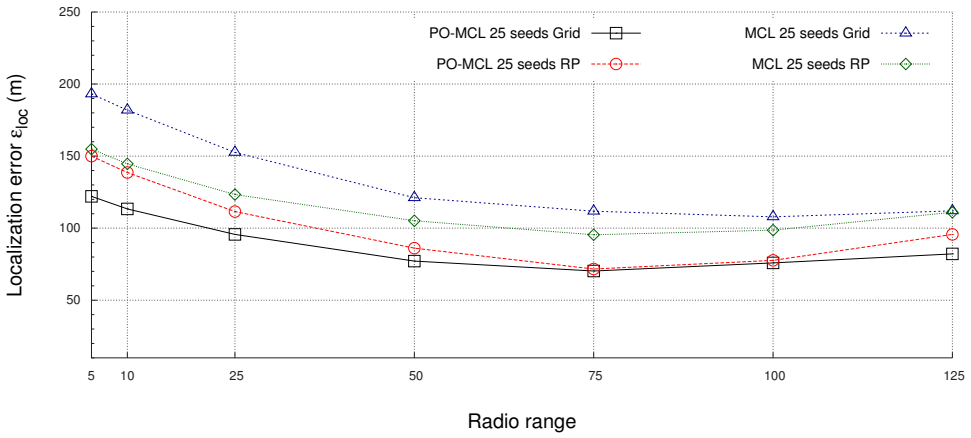


Figure A.6: Localization error ϵ_{loc} for $N_{\text{sample}} = 25$, $magQuery = 1.0$, $N_{\text{seeds}} = 25$

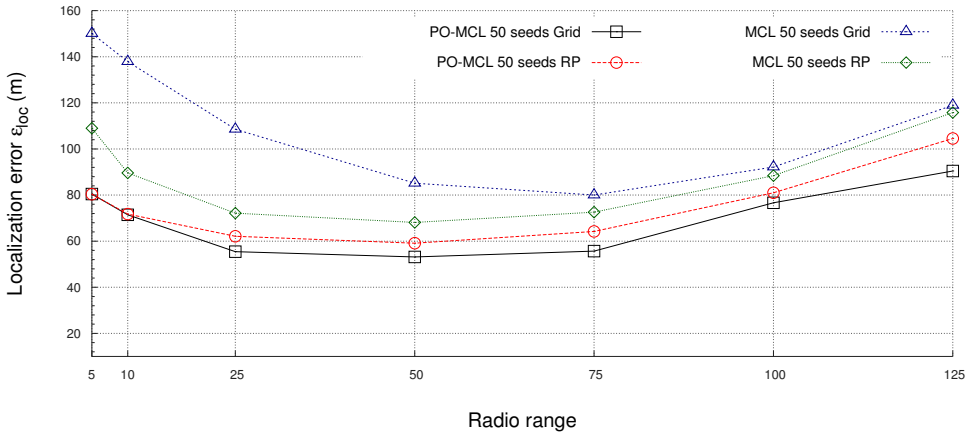


Figure A.7: Localization error ϵ_{loc} for $N_{\text{sample}} = 25$, $magQuery = 1.0$, $N_{\text{seeds}} = 50$

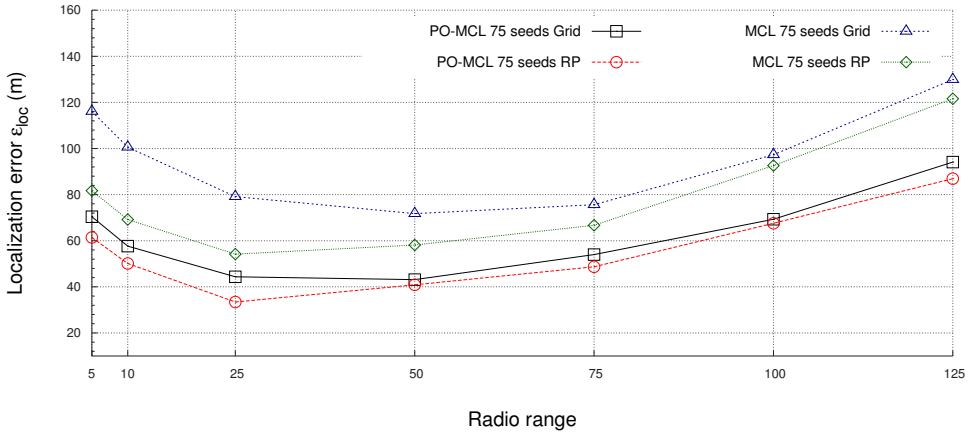


Figure A.8: Localization error ϵ_{loc} for $N_{\text{sample}} = 25$, $magQuery = 1.0$, $N_{\text{seeds}} = 75$

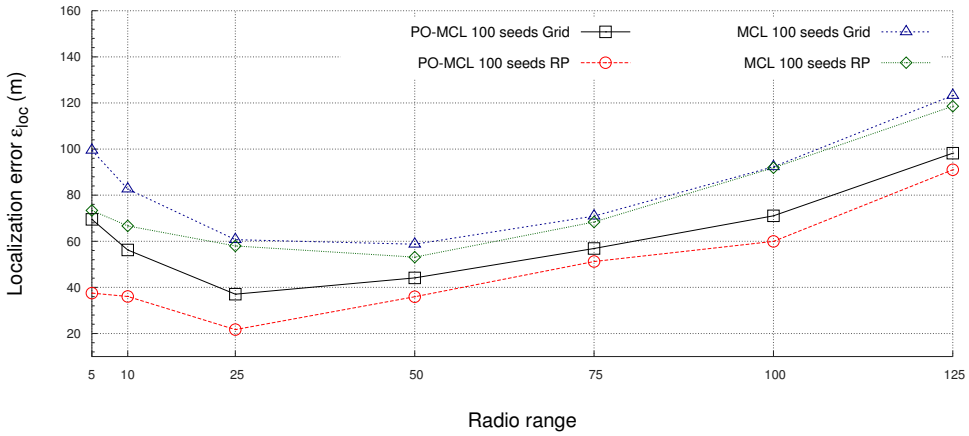


Figure A.9: Localization error ϵ_{loc} for $N_{\text{sample}} = 25$, $magQuery = 1.0$, $N_{\text{seeds}} = 100$

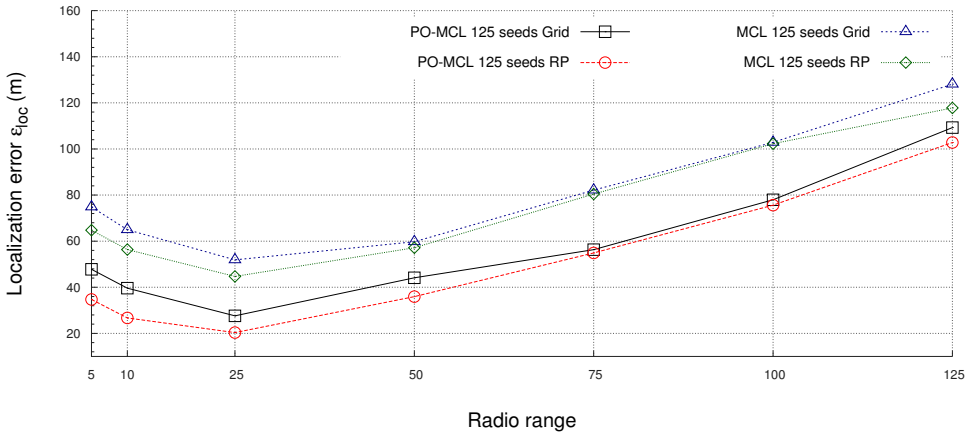


Figure A.10: Localization error ϵ_{loc} for $N_{\text{sample}} = 25$, $magQuery = 1.0$, $N_{\text{seeds}} = 125$

Curriculum Vitae

Salke Hartung

Personal Information

Born	1984-06-04 in Bad Hersfeld
Citizenship	German

Contact Information

Address	Am Ebelhof 7, 37075 Goettingen, Germany
Email	salke.hartung@gmail.com

Education

since 02/2011

PhD Student
PCS Programme in Computer Science (PCS)
Georg-August University School of Science (GAUSS)
Institute of Computer Science, Telematics Group.

- Thesis Topic: Efficient Range-Free Monte Carlo Localization for Mobile Wireless Sensor Networks
- Advisors: Prof. Dr. Dieter Hogrefe, Prof. Dr. Xiaoming Fu

12/2008–08/2009

Study Abroad
University of Wollongong, Australia
Faculty of Informatics

06/2011 Master of Science
Georg-August University Goettingen
Institute of Computer Science, Telematics Group

- Thesis Topic: Anonymous Routing for Mobile Adhoc Networks
- Advisor: Prof. Dr. Dieter Hogrefe
- Thesis Grade: 1.0
- Degree Grade: 1.3, **Graduated with honors**

07/2007 Bachelor of Science
Georg-August University Goettingen
Institute of Computer Science
in cooperation with the Institute of Geography

- Thesis Topic: Implementierung eines WMS Clients zur Integration in ein interdisziplinäres Forschungsprojekt
- Advisor: Dr. Stefan Erasmi
- Thesis Grade: 1.7
- Degree Grade: 1.8 (Specialization: Geoinformatics)

Publications

Sensor-Assisted Monte Carlo Localization for Wireless Sensor Networks. Salke Hartung, Somayeh Taheri, and Dieter Hogrefe. In 6th IEEE International Conference on Cyber Technology (CYBER), Hong Kong, HK, June 2014.

Anonymous Group-Based Routing in MANETs. Somayeh Taheri, **Salke Hartung**, and Dieter Hogrefe. In Journal of Information Security and Applications. Elsevier, 2014.

Sensor-Assisted Monte Carlo Localization for Wireless Sensor Networks. Salke Hartung, Ansgar Kellner, Arne Bochem, and Dieter Hogrefe. In 6th IFIP International Conference on New Technologies, Mobility and Security (NTMS) - Poster + Demo Session, Dubai, UAE, April 2014.

Practical RSSI Long Distance Measurement Evaluation in Wireless Sensor Networks. Salke Hartung, Henrik Brosenne, and Dieter Hogrefe. In The 2013 IEEE Conference on Wireless Sensors (ICWiSe 2013), Kuching, Malaysia, December 2013.

Anonymity and Privacy in Multicast Mobile Ad Hoc Networks. Somayeh Taheri, **Salke Hartung**, and Dieter Hogrefe. In The 6th ACM International Conference on Security of Information and Networks (ACM/SIGSAC SIN 2013), Aksaray, Turkey, November 2013.

RDIS: Destination Location Privacy in MANETs. Somayeh Taheri, **Salke Hartung**, and Dieter Hogrefe. International Journal of Information Privacy, Security and Integrity (IJIPSI), Vol. 1, Nos. 2/3, 2012.

Achieving receiver location privacy in Mobile Ad Hoc Networks. Somayeh Taheri, **Salke Hartung**, and Dieter Hogrefe. In Proceedings of the IEEE International Conference on Information Privacy, Security, Risk and Trust (PASSAT2010), Minneapolis, USA, August 2010.

Peer Review Service

IEEE International Conference on Communications, 2015
(ICC 2015)

EAI International Conference on Bio-inspired Information and Communications Technologies, 2015

Supervised Thesis

Efficient Localization for Mobile Wireless Sensor Networks,
Master Thesis, Arne Bochem, 2015

An ESRI-Shapefile-based Mobility Model Implementation for the Qualnet Network Simulator,
Bachelor Thesis, Andreas Zdziarstek, 2014

Analyse der Sendereichweitebestimmung des Netzwerksimulators Qualnet,
Student project, Andreas Zdziarstek, 2014

Teaching Experience

Teaching Assistant since 2011

- Rechnerarchitektur I, Rechnerarchitektur II
  - Summer Term 2011-2013
  - (Under-)graduate course about computer architecture
  - Responsible for 1 hour weekly exercise sessions

- Rechnernetze I, Rechnernetze II
  - Winter Term 2011/12-2013/14
  - (Under-)graduate course about computer networks
  - Responsible for 1 hour weekly exercise sessions
- Datenbanken
  - Undergraduate course about fundamentals of databases
  - Summer Term 2012-present
  - Responsible for 1.5 hour exam
- SecLab
  - Summer Term 2015
  - Practical lab covering different aspects of computer security
  - Responsible for design of practical lessons about Reverse Code Engineering (RCE)
  - Responsible for assisting students during exercise sessions
- Mobile Communication
  - (Under-)graduate course about mobile telecommunications and ad hoc networking
  - Responsible for 1 hour weekly exercise sessions
  - Responsible for 2 hour exam