Projektbericht

Wireless Sensor Networks-Labor (WSNLAB)

Abschlussdokumentation

Dr. Nils Aschenbruck
Jan Bauer
Jakob Bieling
Alexander Bothe
Matthias Schwamborn
Prof. Dr. Peter Martini

universitätbonn

Dr. Dennis Pfisterer
Kinan Hakim
Prof. Dr. Stefan Fischer

UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR TELEMATIK

Dr. Carsten Buschmann

coalesenses

Frank Gehring
Christian Wieschebrink

Bundesamt
für Sicherheit in der
Informationstechnik

Versionsübersicht

Version	Datum	Beschreibung
1.0	27.06.2011	vollständiger Abschlussbericht
1.1	21.07.2011	Änderungswünsche eingearbeitet

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Drahtlose Sensornetze bestehen aus vielen relativ kleinen, drahtlos vernetzten Geräten, die mit Sensoren ausgestattet sind. Jeder Sensorknoten kann Daten sammeln, verarbeiten und diese weiterleiten. Die Kommunikation erfolgt dabei typischerweise durch multi-hop Kommunikation. Dabei fungieren die Knoten zugleich als Relais. Um einen Knoten zu erreichen, der sich außerhalb der eigenen Sendereichweite befindet, werden die Daten an einen dem Adressaten „näher“ liegenden (Relais-)Knoten übermittelt. Dieser leitet die Daten dann ggf. über weitere (Relais-)Knoten zum Ziel weiter. Gerade in Sensornetzen kann auf den (Relais-)Knoten auch eine verteilte Verarbeitung, Aufbereitung und Fusion der gewonnenen Daten erfolgen.

Typische Einsatzgebiete von Sensornetzen sind Szenarien, in denen Lebewesen, Infrastruktur oder Gebiete beobachtet oder überwacht werden. Da es sich dabei zunehmend auch um sicherheitskritische Bereiche oder sicherheitskritische Szenarien handelt, wird es immer wichtiger, Sensoren sowie Sensornetze vor störenden Einflüssen und bösartigen Angriffen zu schützen. Darüber hinaus sollten auch bei allgemeinen Szenarien Sicherheits- und insbesondere Privacy-Aspekte nicht vernachlässigt werden.

In dem Projekt „Wireless Sensor Network Lab“ (WSNLAB) wurde zunächst ein Konzept zum Aufbau einer Labor-Testumgebung für drahtlose Sensornetze entwickelt [7]. Aufbauend auf dem entwickelten Konzept wurden statische Sensornetze aufgebaut sowie eine Sicherheitsarchitektur entwickelt [6]. Diese wird hinsichtlich ihrer Leistungsfähigkeit untersucht. Anschließend sollen mobile Sensornetze aufgebaut und ebenfalls auf ihre Leistungsfähigkeit und Sicherheit hin untersucht werden. Insbesondere die Auswirkungen möglicher Angriffe und die entwickelte Sicherheitsarchitektur sollen evaluiert werden.

1.2 Beiträge und Struktur dieses Dokuments

Dieses Dokument ist in drei Teile gegliedert. Teil I beschreibt die Testumgebung, wobei zunächst detailliert auf Analyse und Entwurf selbiger eingegangen wird (Kap. 2). Darauf basierend wird die Umsetzung des statischen Testbeds in Kap. 3 beschrieben. Kap. 4 erklärt weiterhin die Erweiterung des Testbeds um Mobilität und schließt Teil I ab. Teil II umfasst Konzept (Kap. 5) und Umsetzung (Kap. 6) der Sicherheitsarchitektur, welche einen wesentlichen Bestandteil

des Projekts ausmacht. Teil III besteht aus den gewonnenen Ergebnissen der Lokalisierungs- und Vitaldatenanwendung. Dazu wird einleitend ein Überblick über bekannte Lokalisierungsverfahren gegeben (Kap. 7). Eine Auswahl dieser Verfahren wird darauffolgend simulativ (Kap. 8) sowie experimentell (Kap. 10) evaluiert. Abgeschlossen wird Teil III durch die Beschreibung der Vitaldatenerweiterung in Kap. 11.

Im Rahmen des Testbed-Konzepts (Kapitel 2) werden zunächst verschiedene Sensorknoten sowie Sensornetz-Betriebssysteme beschrieben. Betriebssysteme werden (sofern sinnvoll und aktuell) prototypisch in Betrieb genommen und deren Leistungsfähigkeit bzgl. diverser Metriken betrachtet. Abschließend wird eine Empfehlung abgegeben, welche Betriebssysteme für das Testbed berücksichtigt werden sollen. Im Anschluss werden für das Konzept nicht zu vernachlässigende Aspekte wie „Routing“, „Automatische Updates und Fernwartbarkeit“ sowie „Topologie-Management im Testbed“ betrachtet. Abschließend erfolgt dann die Beschreibung des Konzepts für den Laboraufbau.

In Kapitel 3 werden zu Beginn alternative Funktechnologien und Sensormethoden geprüft und beschrieben. Anschließend erfolgt ein Überblick über den Laboraufbau, bei dem auch die aktuellen Anwendungen beschrieben und Herausforderungen beim Aufbau erläutert werden.

In Kapitel 4 wird die Mobilitätserweiterung vorgestellt. Dabei werden nach einem Überblick über die Gesamtarchitektur für die Integration von Mobilität ins Testbed die Kernkomponenten detailliert beschrieben. Im Anschluss folgt die Realisierung der Erweiterung.

Die entwickelte Sicherheitsarchitektur (Kapitel 5) wurde so eng wie möglich an das BSI-Projekt MoSe [18] angelehnt. Neben der Bedrohungsanalyse sowie der Beschreibung der Angriffe werden die zur Realisierung im Testbed vorgesehenen Gegenmaßnahmen dargelegt.

Die Umsetzung der Sicherheitsarchitektur (Kapitel 6) beginnt mit der Einführung des Intrusion Detection Systems als zentraler Komponente. Weiter werden Implementierung und Evaluation ausgewählter Angriffe und Gegenmaßnahmen aus der Sicherheitsarchitektur beschrieben. Hierbei werden auch Herausforderungen bei der Implementierung und Ideen für weitere Forschungsarbeiten herausgestellt. Außerdem werden Simulationsergebnisse im Kontext eines Skalierbarkeitstests des Gesamtsystems vorgestellt. Abschließend erfolgt eine Einordnung und Beschreibung von Schutzmaßnahmen gegen unkooperative und egoistische Sensorknoten.

Die theoretischen Grundlagen der Lokalisierung in Sensornetzen werden in Kapitel 7 beschrieben. Es werden verschiedene Techniken zur Lokalisierung vorgestellt und auf ihre Eignung für Sensornetze geprüft. Verfahren, die nach dem Prinzip der Lateration arbeiten, sind in Sensornetzen am besten einsetzbar und werden daher etwas genauer betrachtet. Sie berechnen die Positionen aller Sensorknoten aus den bekannten Positionen einiger weniger Knoten, so genannten Ankern, unter Zuhilfenahme von Distanzschätzungen zu den Ankern. Entspre-

chende Distanzschätztechniken werden daher ebenfalls detailliert beschrieben. Hierbei liegt ein besonderer Schwerpunkt auf solchen Verfahren, die auf Basis der Funkschnittstelle arbeiten.

Ausgesuchte Lokalisierungsverfahren wurden für einen Simulator implementiert und ihre Leistungsfähigkeit simulativ untersucht. Kapitel 8 erläutert zunächst die Entwicklung eines Modells zur Beschreibung der Funksignalabschwächung auf Basis von Messungen in der Realität. Es werden weiterhin die simulierten Szenarien vorgestellt und die Simulationsumgebung beschrieben. Im Anschluss werden die Ergebnisse vorgestellt und diskutiert.

Die im Projekt entwickelte Lokalisierungsanwendung wird in Kapitel 9 vorgestellt. Es wird die Architektur der Anwendung beschrieben, und dann auf die Testumgebung für erste Indoor-Tests eingegangen, die auf dem Wisebed-Testbed durchgeführt wurden.

Die experimentelle Erprobung der Lokalisierungsverfahren wird in Kapitel 10 dargestellt. Es wurden zwei Experimente durchgeführt. Einerseits wurde ein Lokalisierungsexperiment mit 28 Sensorknoten durchgeführt. Hier kam eine Positionsbestimmung mittels Distanzschätzungen auf Basis von Signalstärkenmessungen zum Einsatz. Abschnitt 10.1 beschreibt Aufbau, Durchführung und Ergebnisse des Experimentes. Abschnitt 10.2 hingegen widmet sich den Messungen zur Distanzschätzung auf Basis der Signalumlaufzeit und erläutert Versuchsaufbau und Ergebnisse.

Die Ergebnisse des Arbeitspaketes „Vitaldatensensorik“ werden in Kapitel 11 dargestellt. Der erste Teil des Kapitels beschreibt die entwickelte Hardware, während sich der zweite Teil mit der Treibersoftware für das Modul beschäftigt.

Teil I

Testumgebung

2 Analyse und Entwurf eines Testbeds

Um eine fundierte Entscheidung darüber treffen zu können, welche Software innerhalb des Testbeds zum Einsatz kommen soll, werden zunächst in Abschnitt 2.1 nach einführender Beschreibung der Sensorknoten-Hardware verschiedene Betriebssysteme für Wireless Sensor Networks (WSNs) verglichen und evaluiert. In Abschnitt 2.2 werden daraufhin verschiedene Routing-Protokolle kurz beschrieben und deren Vor- und Nachteile diskutiert. Ein weiterer wichtiger Aspekt der automatischen Updates von Software auf den Sensorknoten wird in Abschnitt 2.3 behandelt. Darauffolgend werden in Abschnitt 2.4 unterschiedliche Ansätze zum Topologie-Management in einem Testbed diskutiert. In Abschnitt 2.5 erfolgt dann die Beschreibung des Testbed-Konzepts. Eine kurze Zusammenfassung der wesentlichen Punkte aus diesem Kapitel wird schließlich in Abschnitt 2.6 gegeben.

2.1 Betriebssysteme und Sensorknoten

Sensorknoten sind stark eingeschränkt durch Prozessorleistung, Arbeitsspeicher, Akkuleistung und Netzwerk-Bandbreite. Daher können Betriebssysteme (Operating Systems (OSs)) für weniger limitierte Hardware-Geräte wie Notebooks, Smartphones oder eingebettete Systeme i.A. nicht für Sensorknoten eingesetzt werden. Stattdessen müssen neue Ansätze beim Design eines OS für Sensorknoten verfolgt werden, so dass die vorhandenen Ressourcen effizient genutzt werden. Zunächst werden in Abschnitt 2.1.1 die besonderen Herausforderungen für ein Sensorknoten-OS aufgezeigt. Im darauffolgenden Abschnitt 2.1.2 wird nach Beschreibung der Sensorknoten-Hardware eine Übersicht aktueller OSs gegeben und anhand praxisorientierter Kriterien eine Vorauswahl zur weiteren Betrachtung getroffen. Die Einsetzbarkeit dieser OSs in verschiedenen Anwendungsbereichen wird in Abschnitt 2.1.3 untersucht. Das Kapitel wird durch ein Fazit in Abschnitt 2.1.4 abgeschlossen.

2.1.1 Herausforderungen

Aufgrund der besonderen Eigenschaften von Sensorknoten und WSNs existieren mehrere Herausforderungen für ein entsprechendes OS [136]. Zum einen erschweren die stark limitierten Ressourcen eines Sensorknotens das Management derselben. Zum anderen wird in Anbetracht der großen Zahl an unterschiedlichen Sensor-Plattformen ein hohes Maß an Portabilität gefordert. Weiterhin müssen sich OSs für WSNs leicht an die Anforderungen des betrachteten

Anwendungsszenarios anpassen lassen. Da Sensorknoten häufig nebenläufige Aufgaben zu erfüllen haben, ist außerdem ein effizientes Ausführungsmodell erforderlich.

Limitierte Ressourcen

Zu den knappen Ressourcen eines Sensorknotens gehören Batterie, Prozessor, Speicher und Bandbreite. Die Werte eines repräsentativen Sensorknotens (TelosB [109]) sind:

- Stromaufnahme CPU: 1.8 mA / 5.1 μA (active/sleep)
- 8 MHz 16-bit RISC MicroController Unit (MCU)
- Stromaufnahme Funkchip: 23 mA / 21 μA / 1 μA (receive/idle/sleep)
- 48 kB Programmspeicher, 10 kB RAM, 1024 kB serieller Flash-Speicher
- 250 kbps Datenrate

Im Folgenden wird genauer auf die einzelnen Ressourcen eingegangen.

Batterie Der überwiegende Teil an WSN-Anwendungsgebieten erfordert eine lange Lebensdauer von mehreren Monaten bis Jahren [139]. Da die Batterien, welche dem Sensorknoten nur eine begrenzte Menge an Energie bereitstellen können, nach Inbetriebnahme des Netzes i.A. nicht ausgetauscht werden, ist *Energieeinsparung* die primäre Herausforderung in einem WSN [3].

Wie schon an obigen Daten zu erkennen ist, verbraucht Kommunikation zwischen Sensorknoten deutlich mehr Energie als lokale Berechnungen oder Sensordatenerfassung. Aber nicht nur das Versenden und Empfangen, sondern auch das Lesen und Schreiben von Daten vom bzw. auf Flash-Speicher erfordert ein erhebliches Maß an Energie [136]. Daher spielt u.a. die *effiziente Umprogrammierung* (vgl. Abschnitt 2.3) von Sensorknoten eine wichtige Rolle: Je weniger Daten für diese benötigt werden, desto weniger Kommunikation und desto weniger Flash-Speicher wird in Anspruch genommen, was folglich den Energiekonsum verringert.

Das OS ist verantwortlich für die Bereitstellung von Energiespar-Mechanismen, wobei periodische Inaktivität durch Übergehen in den *Sleep-Modus* eine dieser Methoden ist. Nach [74] wird zwischen drei Sleep-Modi unterschieden:

- *idle* schaltet nur den Prozessor aus.
- *power down* schaltet alles außer den Watchdog zur Fehlerüberwachung und die asynchrone Interrupt-Logik zum Aufwecken des Knotens aus.
- *power save* ist ähnlich zu power down, lässt zusätzlich jedoch einen asynchronen Timer laufen.

Die Abstraktion des Energiemanagements auf unterer Ebene besteht nun darin, sogenannte Sleep Rates und Duty Cycle Perioden auf Benutzerebene in Abhängigkeit von der entsprechenden Anwendung zu definieren.

Prozessor Der exemplarische TI MSP430 Prozessor des TelosB Sensorknotens hat eine Leistung von wenigen Million Instructions Per Second (MIPS) [161]. Um zu verhindern, dass Prozesse mit hoher Priorität verzögert werden, ist es die Aufgabe des OS-Schedulers, rechenintensive Prozesse zeitlich sinnvoll einzuteilen. Dies ist besonders bei den effizienten *Ereignis-basierten Laufzeit-Modellen*, welche dem run-to-completion Prinzip folgen, eine Herausforderung [136].

Speicher Zu den Haupteinschränkungen von Sensorknoten zählt auch der Programmspeicher [136], welcher auf dem TelosB nicht mehr als 48 kB ausmacht. D.h. das komplette OS sowie benötigte Gerätetreiber und Anwendungen werden durch diese Komponente limitiert und müssen daher so klein wie möglich sein. Jedes von der Anwendung benötigte Modul wird vom Programmspeicher in den flüchtigen RAM-Speicher (10 kB) geladen und ausgeführt. Neben dem Programmspeicher ist auch der serielle Flash-Speicher nicht-flüchtig und kann als allgemeine Datenablage, z.B. zum Logging von gemessenen Sensordaten, verwendet werden.

Bandbreite Sensorknoten in einem WSN kommunizieren drahtlos. Zu den weit verbreiteten Standards für drahtlose Kommunikation zählen Institute of Electrical and Electronics Engineers (IEEE) 802.11 (Wireless Local Area Network (WLAN)), 802.15.1 (Bluetooth) und *802.15.4* (Low-Rate Wireless Personal Area Network (LR-WPAN)). Aufgrund des zu hohen Energieverbrauchs von WLAN- und Bluetooth-Funkchips sind die meisten aktuellen Sensorknoten mit einem Funkchip ausgestattet, der kompatibel zu IEEE 802.15.4 ist [136]. Der geringe Energieverbrauch führt allerdings auch zu einer deutlich geringeren (maximalen) Datenrate von 250 kbps (im Vergleich zu 54 Mbps bzw. 3 Mbps). Die maximale Sendereichweite von LR-WPAN-Antennen liegt bei etwa 100 m.

Portabilität

Um das breite Spektrum an unterschiedlichen Hardware-Plattformen so gut wie möglich abzudecken, muss ein OS für WSNs portabel sein. Dazu gehören neben entsprechenden Treibern z.B. auch Abstraktionen für Entwickler, so dass Anwendungen nicht für jede Plattform angepasst werden müssen. Die Herausforderung dabei ist, das OS so zu programmieren, dass beim Portieren auf eine neue Plattform die nötigen Änderungen am Code so gering wie möglich ausfallen [136].

Adaptierbarkeit

Die Charakteristika einer WSN-Anwendung sind ebenso zahlreich wie die Anwendungen selbst (vgl. [139]). Software für WSNs ist i.A. stark auf die jeweilige Anwendung spezialisiert und muss daher je nach Anwendungstyp unterschiedlichen Anforderungen genügen. Gerade bei Langzeitanwendungen, von denen – wie weiter oben bereits erwähnt – viele im WSN-Bereich anzutreffen sind, kommt es häufiger vor, dass die auf den Sensorknoten laufende Software über die Zeit hinweg geändert wird, weil z.B. ein anderer Fokus bei der Datenerfassung gesetzt wird. Eine Möglichkeit, die Sensorknoten umzuprogrammieren, ist, alle betroffenen Knoten einzusammeln und einzeln per Kabel zu flashen. In Anbetracht der großen Zahl an Knoten und der dadurch verlorenen Zeit zur Datenerfassung ist diese Methode jedoch sehr unpraktikabel. Stattdessen wird eine drahtlose Umprogrammierung verfolgt, welche die Ausfalldauer minimieren soll, indem die geänderte Software durch das Netz propagiert und die eigentliche Umprogrammierung von den Knoten selbständig durchgeführt wird. Dafür muss vom OS ein entsprechendes Framework bereitgestellt werden. Die Herausforderung dabei ist nun, den Overhead vom eingesetzten Verbreitungsprotokoll und der Nutzdaten selbst so gering wie möglich zu halten und dadurch die Energiekosten zu minimieren.

Multi-Tasking

Eine typische Sensor-Anwendung sieht wie folgt aus [136]: Von der unmittelbaren Umgebung werden Sensordaten erfasst, anhand vorher definierter Regeln aggregiert, (möglicherweise) verschlüsselt und an die Netzwerk-Senke via multihop gesendet. Die Aufgaben des Sensorknotens in diesem Beispiel sind demnach im Einzelnen:

- Sensordaten erfassen
- Daten von benachbarten Knoten sammeln
- gesammelte Daten aggregieren
- gesammelte Daten ver- und entschlüsseln
- Daten in Richtung der Senke routen

Da einige der oben genannten Aufgaben nebenläufig sind, ist Multi-Tasking-Fähigkeit eine wichtige Anforderung an ein WSN OS. MCU-gestützte Parallelisierung ist stark beschränkt und der sogenannte Context Switch Overhead, welcher beim Hin- und Herspringen zwischen verschiedenen Tasks entsteht, ist beachtlich. Für das OS besteht die Herausforderung darin, nebenläufige Tasks mit Hilfe eines entsprechenden Laufzeit-Modells effizient in Bezug auf Verzögerung und Energieverbrauch auszuführen.

Plattform	IRIS	iSense	MicaZ	TelosB
Modell	XM2110	Core Module	MPR2400	TPR2420
Frequenz	2.4 GHz	2.4 GHz	2.4 GHz	2.4 GHz
Prozessor	Atmel ATmega1281	Jennic JN5139	Atmel ATmega128L	TI MSP430
Funkchip	RF230 Atmel	Jennic JN5139	TI CC2420	TI CC2420
Serieller Flash	512 kB	128 kB	512 kB	1024 kB
RAM	8 kB	96 kB	4 kB	10 kB

Tabelle 2.1: Hardware-Spezifikation der Sensorplattformen.

	MEMSIC					**iSense**				
	MDA100	**MTS300**	**MTS310**	**MTS400**	**MTS420**	**Environmental**	**GPS**	**Security**	**Vehicle Detection**	**Weather**
Akustik		✓	✓							
Barom. Druck				✓	✓					✓
Beschleunigung			✓					✓		
Buzzer		✓	✓							
Externer Sensor	✓									
GPS					✓		✓			
Infrarot								✓		
Licht	✓	✓	✓	✓	✓	✓				
Luftfeuchtigkeit				✓	✓					✓
Magnet. Feld			✓						✓	
Temperatur	✓	✓	✓	✓	✓	✓				✓

Tabelle 2.2: Unterstützte Sensoren der MEMSIC und iSense Sensorboards.

2.1.2 Überblick

Im Folgenden werden zunächst die Sensorknoten und -boards beschrieben, worauf ein Überblick über in der Literatur vertretene OSs gegeben wird.

Sensorknoten

International am weitesten verbreitet sind die Sensorknoten der Firma MEMSIC (ehemals Crossbow) [107] mit den Plattformen IRIS, MicaZ und TelosB. Weiterhin von Interesse ist die iSense-Plattform der deutschen Firma Coalesenses [38]. Eine detaillierte Spezifikation dieser vier Plattformen ist in Tabelle 2.1 zu sehen. Die TelosB-Plattform verfügt über ein USB-Interface, wodurch der Sensorknoten direkt an einen USB-Port gesteckt und programmiert bzw. getestet werden kann. Zudem sind auf dieser Plattform Sensoren für Temperatur, Licht und Luftfeuchtigkeit direkt integriert. Die Plattformen IRIS, iSense und MicaZ stellen hingegen Basismodule dar, auf denen hauptsächlich Prozessor, Speicher und Funkchip zur Verfügung stehen. Diese müssen aufgrund des fehlenden USB-Interfaces auf ein sogenanntes Gateway gesteckt werden, um sie

OS	Letztes Release	URL
Contiki	16.02.2010 (2.4)	www.sics.se/contiki
CORMOS	—	www.ics.forth.gr/carv/scalable/cormos.html
DCOS	—	www.eyes.eu.org
iSense	13.06.2010 (r3993)	www.coalesenses.de
KOM-OS	—	www.iis.fraunhofer.de/EN/bf/ec/dk/sn/lsg/index.jsp
MantisOS	—	mantis.cs.colorado.edu
Nano-RK	18.06.2010 (r1098)	www.nanork.org
PEEROS	—	www.eyes.eu.org
SenOS	—	redwood.snu.ac.kr
SOS	21.06.2007 (2.0.1)	https://projects.nesl.ucla.edu/public/sos-2x/doc/
TinyOS	06.04.2010 (2.1.1)	www.tinyos.net

Tabelle 2.3: Überblick WSN OSs

mit einem PC verbinden zu können. Weiterhin sind sie modular, d.h. es existieren verschiedene Module (Sensorboards), welche auf das Basismodul gesteckt werden und so (je nach Modul) bestimmte Sensoren anbinden. Tabelle 2.2 listet die verschiedenen Sensorboards auf und gibt an, welche Sensoren unterstützt werden (vgl. [38, 106]).

Betriebssysteme

Nachdem die verschiedenen Herausforderungen für WSN OSs weiter oben bereits diskutiert wurden, wird nun eine Vorauswahl an OSs für die weitere Betrachtung nach folgenden praxisorientierten Kriterien getroffen:

Verfügbarkeit In der Literatur existiert eine breit gefächerte Auswahl an OSs (vgl. z.B. [136]). Allerdings sind die meisten davon lediglich prototypisch implementiert worden und nicht öffentlich verfügbar. Da es im Rahmen dieses Projektes um den praktischen Einsatz in einem Testbed geht, erscheint es sinnvoll, nur OSs weiter zu betrachten, von denen auch eine funktionsfähige Implementierung verfügbar ist.

Entwicklungsstand Ein weiterer wichtiger Faktor ist der Entwicklungsstand eines OS. Es empfiehlt sich, nur Implementierungen, die fortlaufend weiterentwickelt werden, auszuwählen. Somit kann ein gewisses Maß an Qualität erwartet werden, da Fehler bereinigt, Optimierungen vorgenommen und neue Features hinzugefügt werden.

Benutzergemeinde Die Benutzergemeinde einer OS-Implementierung trägt wesentlich dazu bei, dass die Qualität ständig verbessert wird. Dies gelingt, indem z.B. Bug-Reports oder Verbesserungsideen an die Entwickler gesendet werden. Weiterhin sorgt eine große Benutzergemeinde dafür, dass viele Sensor-Plattformen von dem jeweiligen OS unterstützt werden, indem sie entweder selbst die Portierung vornimmt oder eine Anfrage an die Entwickler sendet. Im Allgemeinen ist eine große Benutzergemeinde wünschenswert, da so schneller Fragen geklärt werden können und durch Beisteuerung von Code an der Implementierung mitgewirkt wird.

Eine Übersicht an OSs basierend auf [136] ist in Tabelle 2.3 gegeben. Es sei an dieser Stelle erwähnt, dass das im Lastenheft genannte OS eCos [62] auf eingebettete Systeme, welche mehr Ressourcen zur Verfügung haben, ausgerichtet ist und daher in einem WSN nicht sinnvoll eingesetzt werden kann. Die zweite Tabellenspalte gibt, ausgehend vom Stand der Datenerfassung (28.06.2010), an, wann das letzte Release veröffentlicht wurde, wobei von Contiki und TinyOS jeweils das letzte stabile Release angegeben ist. Die Tabelle zeigt, dass nur für 5 der betrachteten 11 OSs eine Implementierung freigegeben wurde. Weiterhin wurde die Entwicklung an SOS Ende 2008 bereits eingestellt und KOM-OS scheidet aus, da es eine kostenpflichtige Lizenz benötigt. Es verbleiben demnach die OSs Contiki [47], iSense, Nano-RK [54] und TinyOS [74], welche die ersten beiden der oben genannten Kriterien erfüllen. Gemessen an Beiträgen auf der jeweiligen Mailingliste sind die Benutzergemeinden von Contiki und TinyOS mit Abstand am größten. Da iSense speziell für die iSense-Plattform entwickelt wurde und bisher keine anderen Sensor-Plattformen wie z.B. die international weit verbreiteten TelosB oder MicaZ unterstützt, ist die Benutzergemeinde eher klein. Nano-RK unterstützt zwar mehrere Plattformen, scheint in Anbetracht der verschwindend geringen Forenbeiträge jedoch nicht weit verbreitet zu sein.

Im Folgenden werden also die OSs Contiki, iSense, Nano-RK und TinyOS weiter untersucht.

MAC-Protokolle In der Literatur sind viele Media Access Control (MAC)-Protokolle für WSNs zu finden (vgl. z.B. [43, 167]). Eine umfassende Übersicht dieser Protokolle würde allerdings deutlich über den Rahmen dieses Berichts hinausgehen. Daher werden nachfolgend lediglich die meist genutzten Protokolle kurz beschrieben. Aussagen bzgl. Verfügbarkeit von Implementierungen beziehen sich auf den Stand der Datenerfassung (08/2010).

Der Standard *IEEE 802.15.4* [14, 82] definiert neben der physikalischen Schicht (PHY) auch die MAC-Schicht. Diese unterscheidet grundsätzlich zwischen zwei Knotentypen, nämlich den Reduced-Function Devices (RFDs) und den Full-Function Devices (FFDs). Letztere sind voll funktionsfähige Knoten, die sowohl als Netzwerk-Koordinator als auch als Endgerät agieren können. RFDs hingegen können nur als Endgeräte dienen und dürfen nur mit einem einzigen FFD interagieren.

IEEE 802.15.4 sieht zwei LR-WPAN-Topologien vor: Zum einen die Stern-Topologie, wobei ein FFD als Personal Area Network (PAN)-Koordinator (Master) fungiert und die anderen Knoten (RFDs oder FFDs, Slaves) nur mit diesem Koordinator kommunizieren. Zum anderen die Peer-to-Peer (P2P)-Topologie, wobei ein FFD mit anderen FFDs über einen oder mehrere Hops kommunizieren kann. Um dieses P2P-Netz zu administrieren, wird zusätzlich einer der FFDs als PAN-Koordinator ausgewählt.

Ein PAN wird mit oder ohne eines sogenannten Superframes koordiniert. Dieser Superframe beginnt mit einem Beacon, welches der Synchronisation dient und Kontrollinformationen für das PAN enthält. Weiterhin ist der Superframe

in eine aktive und inaktive Phase unterteilt, wobei letztere zum Sparen der Energie durch Versetzen in den Schlaf-Modus genutzt wird. Die aktive Phase ist wiederum in die Contention Access Period (CAP), in der ein slotted Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) Protokoll eingesetzt wird, und Contention Free Period (CFP), in der Zeitslots vom Koordinator den einzelnen Knoten fest zugewiesen werden, unterteilt. Endgeräte sparen Energie, da sie die meiste Zeit über im Schlaf-Modus verweilen und regelmäßig aufwachen, um herauszufinden, ob der Koordinator Pakete für sie bereit hält, indem sie auf das Beacon warten.

Wird das PAN nicht durch ein Superframe koordiniert, so wird anstelle des Beacon-Ansatzes ein unslotted CSMA/CA Verfahren verwendet. Der PAN-Koordinator ist permanent erreichbar und bereit, Pakete von den Endgeräten zu empfangen. Endgeräte schlafen hingegen die meiste Zeit und müssen den Koordinator in regelmäßigen Abständen für evtl. ausstehende Pakete pollen. Dieser versendet darauf die Pakete oder signalisiert, dass keine Pakete für das Endgerät vorhanden sind.

Nach bestem Kenntnisstand der Autoren dieses Berichts bietet keines der vier OSs eine Implementierung von obigem IEEE 802.15.4 MAC-Protokoll. Contiki sowie TinyOS setzen derzeit lediglich das Framing um, d.h. die MAC-Kontrolldaten werden in einen 802.15.4-konformen Header geschrieben.

B-MAC [130] ist ein weiteres Carrier Sense Multiple Access (CSMA)-Protokoll und basiert auf der Idee, dass einige Funktionalitäten nicht durch das MAC-Protokoll, sondern durch darüberliegende Protokolle oder Services kontrolliert werden können. Dadurch bleibt B-MAC klein und kann trotzdem dynamisch an die aktuelle Netzlast angepasst werden. Kollisionen werden durch den Clear Channel Assessment (CCA) Algorithmus verhindert. Bei diesem werden Samples der Signalstärke genommen, um das Grundrauschen zu schätzen. Diese Schätzung hilft dann, um entscheiden zu können, ob der drahtlose Kanal beim Abtasten frei oder belegt ist. Backoff-Zeiten vor der Übertragung bestimmt der Service, welcher das Paket an B-MAC gesendet hat, wodurch Fairness und Durchsatz gesteuert werden. Zuverlässigkeit wird durch Acknowledgement (ACK)-Pakete auf Link-Ebene und Energieeffizienz durch Duty-Cycling mittels Low Power Listening (LPL) gewährleistet (für Details, siehe [130]). Implementierungen von B-MAC sind z.Z. vorhanden für Nano-RK und TinyOS. Das Protokoll wird standardmäßig bei Nano-RK eingesetzt.

X-MAC [26] gehört wie B-MAC zu den asynchronen Duty-Cycle-zentrierten MAC-Protokollen. Es verbessert das LPL von B-MAC durch eine kürzere und modifizierte Präambel: Zum einen wird die Empfängeradresse in die Präambel eingebettet, sodass andere Knoten nicht die komplette Präambel abwarten müssen, um herauszufinden, dass das Paket nicht für sie bestimmt ist. Somit können diese sofort wieder in den Schlaf-Modus gehen und sparen weitere Energie. Zum zweiten kann es vorkommen, dass der Empfänger schon zu Beginn einer langen Präambel aufwacht. In diesem Fall muss der Datenaustausch warten, bis die Präambel beendet ist und so verschwenden Sender und Empfänger wertvol-

le Energie. Die Latenz pro Hop ist daher auch durch die Länge der Präambel nach unten beschränkt, was sich bei Multi-hop-Kommunikation signifikant akkumuliert. Als Lösung sieht X-MAC anstelle einer langen Präambel mehrere, durch Pausen unterteilte, kurze Präambeln vor. So kann der Empfänger dem Sender vorzeitig per ACK signalisieren, dass er die Präambel gehört hat. Eine Implementierung des X-MAC Protokolls liegt Contiki bei und wird dort als Standardprotokoll auf der TelosB-Plattform eingesetzt.

Low Power Probing (LPP) [112] ist eine Technik, die darauf ausgelegt ist, ein gesamtes Netz im Vergleich zu einem einzelnen Empfänger (wie bei LPL) aufwachen zu lassen. Dazu wird passives (LPL) durch aktives Abtasten des Kanals ersetzt: Jeder Knoten im Netzwerk versendet in regelmäßigen Abständen ein leeres Paket per Broadcast, mit dem lediglich ein ACK angefordert wird. Falls ein solches ACK empfangen wird, bleibt der Knoten wach und antwortet auf ACK-Anfragen anderer Knoten. Auf diese Weise wachen nach einer gewissen Zeit alle Knoten des Netzwerks auf. Die Prozedur wird durch ein Gateway initiiert, welches permanent wach ist und auf Anfragen reagiert. Eine Implementierung von LPP ist Teil des Contiki OS und ist dort Standard auf der MicaZ-Plattform.

Eine Kombination aus B- und X-MAC (genannt BoX-MAC) liegt der TinyOS-Distribution bei. Als Standard wird allerdings ein simples CSMA-basiertes Protokoll verwendet, ebenso wie bei dem iSense OS.

S-MAC [181] ist CSMA-basiert und versucht, die Energieverschwendung durch die vier Hauptquellen Kollision, Overhearing, Kontrollpaket-Overhead und Idle-Listening zu minimieren. Dazu sieht das Protokoll hauptsächlich drei Komponenten vor: Zum einen werden periodische Abtast- und Schlaf-Phasen verwendet, d.h. der Funkchip wird in regelmäßigen Abständen ein- bzw. ausgeschaltet. Dies führt zu einem festen Duty-Cycle und vermindert das Idle-Listening, also die Zeitspanne, in welcher der Funkchip eingeschaltet ist, obwohl gar keine Kommunikation in der Umgebung stattfindet. Knoten können die eigene Einteilung der beiden Phasen selbst bestimmen und teilen diese der Nachbarschaft per Broadcast mit. Damit wird sichergestellt, dass benachbarte Knoten miteinander kommunizieren können, selbst wenn die Einteilung nicht übereinstimmt. Um Kontroll-Overhead gering zu halten, werden virtuelle Cluster gebildet, in denen alle Knoten bzgl. ihrer Phasen-Einteilung synchronisiert sind. Die zweite Komponente dient der Vermeidung von Kollisionen und Overhearing und setzt im Wesentlichen das aus IEEE 802.11 bekannte CSMA/CA mit Request To Send - Clear To Send (RTS/CTS) um. Das Overhearing-Problem wird dabei umgangen, indem Knoten, welche ein RTS- oder CTS-Paket mithören, in den Schlaf-Modus versetzt werden. Die dritte Komponente ist das sogenannte Message Passing, d.h. eine lange Nachricht wird in mehrere Fragmente unterteilt und als Burst versendet. Für die komplette Nachricht wird nur ein RTS/CTS-Austausch benötigt, wodurch weitere Kontrollpakete eingespart werden. Implementierungen von S-MAC für aktuelle OSs sind derzeit keine bekannt.

T-MAC [171] erweitert das S-MAC Protokoll im Wesentlichen um die Idee, den festen Duty-Cycle durch einen adaptiven zu ersetzen. Die Länge der Abtast-

Phase kann in S-MAC so gewählt werden, dass ein Knoten all seine Nachrichten in dieser aktiven Phase verschicken kann. Da die Nachrichtenrate aber i.A. variiert, kann weitere Energie gespart werden, indem die Länge entsprechend angepasst wird. T-MAC verschickt daher alle Nachrichten als Bursts variabler Länge, zwischen denen die Schlaf-Phase abläuft. Letztere wird allerdings erst dann eingeleitet, wenn für eine bestimmte Zeit kein Event eintritt, welcher die aktive Phase verlängern würde. Auch für T-MAC gilt, dass für aktuelle OSs derzeit keine Implementierungen bekannt sind.

Crankshaft [68] ist ein MAC-Protokoll, das speziell für den Einsatz in engmaschigen Netzen entworfen wurde. Da die Anzahl der Nachbarn in diesen Netzen relativ groß ist, wird z.B. die Energieverschwendung durch Overhearing noch weiter verschärft. Außerdem sind Nachbarschaftsinformationen problematisch, da sie kostbaren RAM belegen. Die grundlegende Idee von Crankshaft ist, dass Knoten nur zu fixen Slots innerhalb eines Frames zum Datenempfang wach sind. Dadurch wird die Zahl der Knoten, die Nachrichten mithören, obwohl sie nicht Empfänger sind, deutlich reduziert. Slots werden unterteilt in Broadcast- und Unicast-Slots, wobei während eines Broadcast-Slots alle Knoten aufwachen. Jeder Knoten horcht zudem während eines Unicast-Slots pro Frame auf dem Medium für eingehende Nachrichten von benachbarten Knoten. Dieser Slot wird in Abhängigkeit von der MAC-Adresse des jeweiligen Knotens bestimmt. Somit ist einem Sender der Unicast-Slot des Empfängers genaustens bekannt. Eine Ausnahme bilden Senke-Knoten: Diese empfangen die meisten Nachrichten und sind daher während aller Unicast-Slots wach. Overhearing kann bei Crankshaft weiterhin durch einen Adressfilter verringert werden. Dieser schaltet den Funkchip aus, sobald das Empfängerfeld des Datenheaders gelesen wurde und die Empfängeradresse nicht mit der Adresse des mithörenden Knotens übereinstimmt. Halkes et al. [68] erwähnen zwar eine TinyOS-Implementierung von Crankshaft, haben diese aber nach bestem Kenntnisstand nicht veröffentlicht.

2.1.3 Evaluation

In diesem Abschnitt werden die oben ausgewählten OSs in Bezug auf weitere praktische Aspekte überprüft. Dazu wird zunächst die Unterstützung verschiedener Sensor-Plattformen verifiziert. Danach wird die Leistungsfähigkeit der einzelnen OSs bzgl. verschiedener Metriken untersucht. Abschließend wird die Interoperabilität der OSs in Kombination mit verschiedenen Sensorknoten getestet.

Plattform-Unterstützung

Ein WSN OS kann nur eingesetzt werden, wenn es auch die verwendeten Sensor-Plattformen unterstützt. Daher wurden die ausgewählten OS zunächst auf Plattform-Unterstützung hin überprüft, d.h. die Angaben der Entwickler wurden verifiziert. Hierzu wurden die untersuchten Sensor-Plattformen (IRIS, iSense, MicaZ und TelosB) jeweils mit einer simplen Beispielanwendung pro-

	IRIS	iSense	MicaZ	TelosB
Contiki			✓	✓
iSense		✓		
Nano-RK			✓	(✓)
TinyOS	✓		✓	✓

Tabelle 2.4: Plattform-Unterstützung relevanter OSs ('✓' bedeutet volle, '(✓)' experimentelle Unterstützung).

	MDA100	MTS300	MTS310	MTS400	MTS420
Contiki		✓	✓		
TinyOS	✓	✓	✓	✓	✓

Tabelle 2.5: Sensorboard-Unterstützung relevanter OSs.

grammiert. Diese dient dazu, grundlegende Funktionsweisen, wie z.B. Aufblinken der LEDs oder Textausgabe auf der seriellen Schnittstelle, zu testen und wird im Sourcecode bereits mitgeliefert. Eine Zusammenfassung der Ergebnisse dieser Tests ist in Tabelle 2.4 aufgeführt. Wie bereits oben erwähnt, wurde iSense speziell für die iSense-Plattform entwickelt und unterstützt bisher somit ausschließlich selbige. Anders herum ist die iSense-Plattform nicht weit genug verbreitet für die Unterstützung durch die anderen betrachteten OSs. Solange also keins der anderen OSs auf die iSense-Plattform portiert wird, muss das iSense OS weiterhin betrachtet werden. TinyOS hingegen unterstützt alle anderen untersuchten Plattformen (IRIS, MicaZ und TelosB) und schneidet damit am besten ab. Contiki unterstützt die beiden weit verbreiteten Plattformen MicaZ und TelosB. Vergleichsweise schlecht schneidet Nano-RK ab, denn es unterstützt lediglich MicaZ-Sensorknoten. Die TelosB-Unterstützung ist zum Zeitpunkt der Datenerfassung (08/2010) lediglich experimentell und scheitert bereits an der Kompilierung.

Die Plattformen IRIS und MicaZ sind modular, d.h. es existieren sogenannte Sensorboards, die sich auf das Basis-Modul stecken lassen (vgl. Überblick Sensorknoten). Tabelle 2.5 zeigt, welche Sensorboards von Contiki und TinyOS z.Z. unterstützt werden, d.h. zu welchen dieser Boards derzeit Programmier-Schnittstellen verfügbar sind. Nano-RK ist nicht in dieser Tabelle vertreten, da keine weiteren Informationen bzgl. der Unterstützung gefunden werden konnten.

Leistungsfähigkeit

Im Folgenden werden die OSs in Bezug auf diverse Metriken evaluiert:

- Datenrate
- Reichweite
- Störanfälligkeit
- Multi-hop-Fähigkeit
- Skalierbarkeit

Contiki	iSense	Nano-RK	TinyOS
95	116	116	114

Tabelle 2.6: Maximale Nutzlast pro Paket (in Bytes)

- Positionierung
- Leistungsaufnahme

Reichweite sowie Störanfälligkeit (in Bezug auf Kommunikation) sind in erster Linie abhängig vom Funkchip, d.h. von der Sensor-Hardware. Multi-hop-Fähigkeit und Skalierbarkeit werden bzgl. der eingesetzten Routing-Protokolle gemessen. Zum Vergleich der OSs verbleiben demnach Datenrate, Positionierung und Leistungsaufnahme. Die „Datenrate“ wird im Folgenden als Durchsatz, also die gemittelte Anzahl an Daten pro Zeiteinheit, angenommen.

Durchsatz Um den Durchsatz messen zu können, wurde für jedes der vier OSs eine simple Test-Anwendung mit Sender- und Empfänger-Seite geschrieben. Der Sender ist hierbei saturiert und sendet daher ununterbrochen Pakete maximaler Größe (128 Bytes) an den Empfänger. Dieser zählt die Anzahl der Pakete, die innerhalb eines vorher definierten Zeitintervalls empfangen wurden. Die Test-Anwendung wurde – soweit möglich – für alle OSs gleich implementiert, um vergleichbare Werte zu erhalten.

Der Durchsatz wird auf Anwendungsebene gemessen, d.h. der Daten-Overhead durch untere Open Systems Interconnection (OSI)-Schichten fließt nicht in die Berechnung mit ein. Dies ist dadurch motiviert, dass der Anwender in erster Linie ohnehin nur an den Nutzdaten interessiert ist.

Die Durchsatz-Messung wurde im WSN-Labor durchgeführt, wobei zwischen Sender- und Empfängerknoten bei einer Distanz von 3 m eine direkte Sichtlinie gegeben war. Weiterhin wurde sichergestellt, dass keine weiteren Sensorknoten eingeschaltet waren, um eine eventuelle Störung des Funksignals zu verhindern. Neben diesen Einflüssen sind außerdem folgende Parameter relevant:

- *MAC-Protokoll*: Das Medienzugriffsprotokoll regelt die Nutzung des gemeinsam verwendeten drahtlosen Mediums und wirkt sich daher auf die Frequenz von Paketversand und -empfang aus.
- *Channel Check Rate*: Dieser MAC-Protokoll-Parameter gibt an, wie häufig der drahtlose Kanal auf Nutzung geprüft wird. Besonders sinnvoll im Kontext von WSNs, wird dieser auf einen kleinen Wert gesetzt, um kostbare Energie zu sparen. Dies führt allerdings auch zu weniger Paketen, die pro Zeiteinheit gesendet bzw. empfangen werden können.
- *Maximale Nutzlast*: Da der Paket-Overhead durch untere OSI-Schichten vom jeweiligen OS bestimmt wird, unterscheidet sich auch die maximale Nutzlast (siehe dazu Tabelle 2.6). Da nur diese für die Berechnung verwendet wird, hat dies auch einen direkten Einfluss auf die Ergebnisse der Durchsatz-Messung.

	Contiki	iSense	Nano-RK	TinyOS
MAC-Protokoll	LPP	CSMA-basiert	B-MAC	CSMA-basiert
Channel Check Rate	64 Hz	—	50 Hz	—

Tabelle 2.7: MAC-Parameter für Durchsatz-Messungen

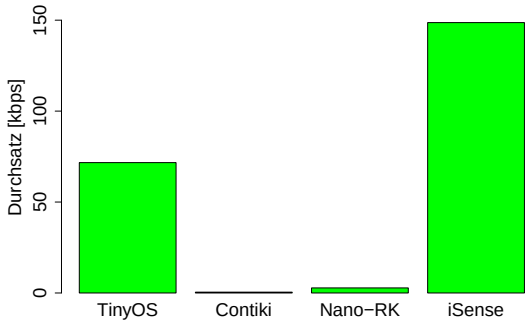


Abbildung 2.1: Durchsatz auf MicaZ- und iSense-Plattform

- *Sample-Intervall*: Dieser Parameter bestimmt, über welche Zeitspanne die Werte gemittelt werden. Je größer dieser Wert, desto geringer sind i.A. die Schwankungen zwischen den Samples einer Messung. Für die durchgeführte Messung wurde ein Wert von 5 s verwendet.

Um möglichst vergleichbare Ergebnisse zu erzielen, wurde gemäß Tabelle 2.4 für Contiki, Nano-RK und TinyOS die MicaZ-Plattform gewählt. Da z.Z. kein MAC-Protokoll auf allen OSs läuft, wurde jeweils das Standard-MAC-Protokoll verwendet. Weiterhin erschien es sinnvoll, die Channel Check Rate auf das Maximum zu setzen, um das drahtlose Medium auslasten zu können (für Details, siehe Tabelle 2.7).

Die Ergebnisse der Messung sind in Abbildung 2.1 zu sehen. Der Barplot gibt jeweils den Median von 100 Samples an. Da die Abweichungen der Samples untereinander verschwindend gering sind, wurde auf eine Darstellung mittels eines Boxplots verzichtet. TinyOS erreicht einen Durchsatz von ca. 72 kbps, während iSense auf der gleichnamigen Plattform 149 kbps erzielt.

Überraschend sind die Ergebnisse für Contiki (~ 0.5 kbps) und Nano-RK (~ 3 kbps). Trotz einer Channel Check Rate von 50 Hz im Fall von Nano-RK wurden nur $3-4$ Pakete/s versendet und empfangen. Bei Contiki wurden zwar ca. 33 Pakete/s versendet, jedoch nur $0-1$ empfangen. Eine weitere Messung, bei der Contiki in Kombination mit dem sogenannten „NullMAC“-Protokoll getestet wurde, hat 230 versendete sowie empfange Pakete/s (~ 175 kbps) ergeben. Da „NullMAC“ die Pakete nur an die angrenzenden OSI-Ebenen weiterreicht, lässt sich daraus schließen, dass die Implementierung des LPP-Protokolls noch nicht stabil ist.

Algorithmus 1 : Pseudocode der Anwendung zum Test des Energieverbrauchs

```
   Input : seq_max, n ∈ ℕ \ {0}
 1 for seq ← 0 to seq_max − 1 do
 2     for i ← 0 to n − 1 do
 3         sleep(1s);
           /* read value from sensor */
 4         ledOn(RED);
 5         val_i ← getTemp();
 6         ledOff(RED);
       /* calculate aggregated values */
 7     val_min ← min_{j∈{0,...,n−1}} val_j;
 8     val_max ← max_{j∈{0,...,n−1}} val_j;
 9     val_avg ← Σ_{j=0}^{n−1} val_j/n;
10     p ← (seq, val_0, ..., val_{n−1}, val_min, val_max, val_avg);
       /* use flash memory */
11     ledOn(YELLOW);
12     addr ← writeToFlash(p);
13     p ← readFromFlash(addr, sizeOf(p));
14     ledOff(YELLOW);
       /* send data */
15     ledOn(GREEN);
16     sendBroadcast(p);
17     ledOff(GREEN);
```

Zusammenfassend lässt sich also feststellen, dass der Durchsatz hauptsächlich von den MAC-Protokollen und deren Implementierungen abhängt und damit einen Vergleich der OSs nur bedingt zulässt.

Positionierung Zum Zeitpunkt der Fertigstellung dieses Berichts (06/2011) sind keine Implementierungen von Lokalisierungsverfahren für irgendeins der betrachteten OSs bekannt. Insbesondere sind keine entsprechenden Algorithmen (siehe Kapitel 7) standardmäßig in den OSs enthalten.

Energieverbrauch Die letzte Metrik, die in diesem Abschnitt betrachtet wird, ist die Leistungsaufnahme, auch Energieverbrauch genannt. Um auch hier vergleichbare Werte zu erhalten, wurde der Energieverbrauch simulativ berechnet. Dazu wurde Avrora [164], ein WSN-Simulator für AVR-Chipsatz-basierte Sensorknoten, in der Version 1.7.111 verwendet. Ursprünglich für die Mica2-Plattform entwickelt, unterstützt dieser Simulator durch die Erweiterung AvroraZ [42] mittlerweile auch MicaZ-Knoten. Das integrierte Modell zur Berechnung des Energieverbrauchs [96] bietet eine Aufschlüsselung nach CPU, Funkchip, LEDs, etc.

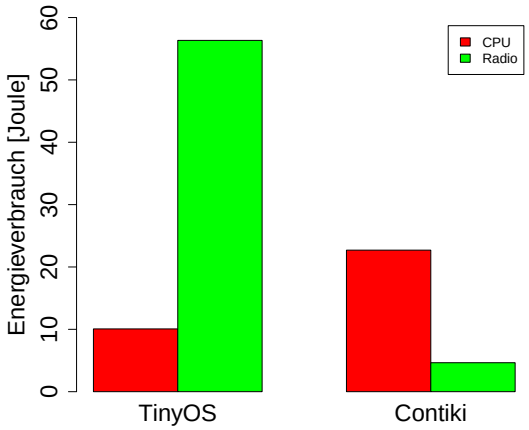


Abbildung 2.2: Energieverbrauch auf MicaZ-Plattform

Für alle vier OSs wurde Algorithmus 1 implementiert. Dieser stellt eine Art Benchmark-Anwendung für Energieverbrauch dar, da er alle relevanten Hardware-Bauteile beansprucht: Jede Sekunde wird ein Sensor-Wert (in diesem Fall der Temperatur-Wert) ausgelesen. Nachdem n Samples ausgelesen wurden, aggregiert der Algorithmus diese Samples zu Minimum, Maximum und Mittelwert. Alle n Samples, aggregierten Werte und eine Sequenznummer (seq) werden darauf zu einem Paket kombiniert und in den Flash-Speicher geschrieben. Diese Daten werden wieder vom Flash-Speicher gelesen und per Broadcast verschickt. Das Auslesen des Sensorwertes, der Zugriff auf den Flash-Speicher und das Versenden der Daten wird jeweils zusätzlich durch Aufblinken eines LEDs begleitet.

Während der Implementierung haben sich mehrere Probleme herausgestellt: Zum einen lassen sich unter dem OS Nano-RK keine Sensorwerte der MicaZ-Sensorboards auslesen (bestätigt durch einen der Autoren von Nano-RK), d.h. die charakteristische und wichtigste Funktion einer Sensor-Plattform im Allgemeinen und dieser im Besonderen wird nicht unterstützt. Da somit keine weitere innerhalb dieses Projekts betrachtete Plattform unterstützt wird, scheidet Nano-RK ebenfalls aus und wird nicht weiter evaluiert. Bzgl. Contiki wurde festgestellt (und durch den Autor der Portierung bestätigt), dass der Flash-Speicher der MicaZ-Plattform derzeit nicht genutzt werden kann, aber auf TelosB-Knoten unterstützt wird. Die TelosB-Unterstützung durch Avrora wiederrum ist jedoch noch experimentell. Daher wurde der Energieverbrauch durch den Flash-Speicher während der Evaluation ausgeblendet. Ebenfalls ausgeblendet wurde das iSense OS, da es für die iSense-Plattform keine Möglichkeit gab, den Energieverbrauch auf vergleichbarer Ebene (simulativ) zu bestimmen.

Die für die Evaluation verwendete Simulationszeit von 1000 s entspricht bei $n = 10$ Samples ungefähr dem Versand von $seq_{max} = 100$ Paketen. Wie erwartet, beanspruchten CPU und Funkchip die meiste Energie und sind daher als einzige Komponenten in Abbildung 2.2 aufgeführt. In dieser ist zu beobachten,

		IRIS	iSense	MicaZ		TelosB	
		TinyOS	iSense	Contiki	TinyOS	Contiki	TinyOS
IRIS	TinyOS	✓			✓		✓
iSense	iSense		✓				
MicaZ	Contiki			✓		✓	
	TinyOS	✓			✓		✓
TelosB	Contiki			✓		✓	
	TinyOS	✓			✓		✓

Tabelle 2.8: Interoperabilität zwischen Plattformen und OSs.

dass einerseits Contiki etwa doppelt so viel Energie durch die CPU verbraucht. Dies ist darauf zurückzuführen, dass bei TinyOS die CPU lange Zeit im idle-Zustand war, während diese bei Contiki permanent aktiv war. TinyOS versetzt die CPU also standardmäßig in den idle-Zustand, während Contiki diese Anweisung explizit von der jeweiligen Anwendung erwartet. Die zweite Beobachtung ist, dass der Funkchip im Fall von TinyOS etwa 12 mal so viel Energie benötigt wie bei Contiki. Dies hängt mit der Channel Check Rate zusammen, welche bei TinyOS ausgeschaltet ist, sodass permanent auf dem drahtlosen Medium gehorcht wird. Insgesamt verbraucht TinyOS in diesem Beispiel zwar deutlich mehr Energie, aber da die Implementierungen nicht bzgl. Energiesparfunktionen optimiert wurden, sollte dieses Ergebnis nicht zu stark bewertet werden.

Interoperabilität

Um die Interoperabilität zwischen den einzelnen Sensor-Plattformen sowie OSs zu testen, wurde eine simple Anwendung verwendet, welche in regelmäßigen Abständen eine „Hello“-Nachricht per Broadcast verschickt. Wird diese Nachricht vom anderen Knoten korrekt empfangen, so werden Sender und Empfänger als interoperabel angesehen. Da es sich hier nur um Single-hop-Kommunikation handelt und alle verwendeten Sensor-Plattformen einen IEEE 802.15.4 kompatiblen Funkchip haben, ist das MAC-Protokoll der entscheidende Faktor. Die Ergebnisse der Tests sind in Tabelle 2.8 dargestellt. Ein „✓“ bedeutet, dass die Kombination der OSs (erste Zeile bzw. Spalte) und Plattformen (zweite Zeile bzw. Spalte) interoperabel ist. Hierzu sei angemerkt, dass nach einem erfolgreichen Test keine anderen (soweit vorhandenen) MAC-Protokolle verwendet wurden.

Wie bereits weiter oben erwähnt, existiert z.Z. kein einheitliches MAC-Protokoll für alle OSs. Daher sind die OSs inkompatibel zueinander, d.h. ein Sensorknoten, auf dem eines der drei OSs läuft, kann derzeit nicht mit einem Sensorknoten kommunizieren, auf dem eines der anderen OSs läuft. Laut [53] ist es möglich, TinyOS-Knoten mit Contiki-Knoten durch Modifikation des Contiki-Sourcecodes kommunizieren zu lassen. Dies konnte jedoch nicht verifiziert werden, da (auch nach Anfrage) der hierfür notwendige Sourcecode nicht zur Verfügung gestellt wurde. Insbesondere sind derzeit keine Implementierungen des IEEE 802.15.4 MAC-Protokolls bekannt und daher konnten selbige auch nicht auf Interoperabilität geprüft werden.

Die Interoperabilität beschränkt sich also auf die Sensor-Plattformen bei fixem OS: Alle unterstützten Plattformen von TinyOS (IRIS, MicaZ und TelosB) sind untereinander interoperabel, solange TinyOS auf Sender und Empfänger verwendet wird. Gleiches gilt für Contiki und die Plattformen MicaZ und TelosB. Folglich ist derzeit zwar ein Plattform-heterogenes, aber kein (uneingeschränktes) OS-heterogenes Testbed realisierbar. Eingeschränkt ist letzteres mit Hilfe von Gateways möglich, welche zwischen unterschiedliche OS-homogene Netze geschaltet werden.

2.1.4 Fazit

In diesem Abschnitt wurde nach Beschreibung der Herausforderungen ein Überblick über verfügbare Hardware und vorhandene Betriebssysteme für Wireless Sensor Networks gegeben. Eine Untersuchung dieser Betriebssysteme anhand praxisorientierter Kriterien, nämlich Verfügbarkeit, Entwicklungsstand und Größe der Benutzergemeinde, hat ergeben, dass lediglich vier der insgesamt elf betrachteten Betriebssysteme für den Einsatz im Testbed in Frage kommen. Diese vier wurden daraufhin bzgl. Plattform-Unterstützung geprüft und anhand verschiedener Metriken evaluiert. Da eins der Betriebssysteme keine der relevanten Sensor-Plattformen vollständig unterstützt, verbleiben die Betriebssysteme Contiki, iSense und TinyOS. Die Evaluation hat deutlich gemacht, dass sich die Betriebssysteme nur auf oberflächlicher Ebene vergleichen lassen, da keine einheitliche Basis der implementierten Protokolle und Parameter existiert. Weiterhin lässt der Sonderfall iSense keinen Vergleich zu Contiki oder TinyOS zu, da dieses Betriebssystem bisher lediglich auf der gleichnamigen Plattform läuft. TinyOS hat einen klaren Vorteil bzgl. Plattform-Unterstützung, da Contiki die IRIS-Plattform derzeit nicht unterstützt. Contiki hingegen ist häufig Vorreiter von neuen Trends und Entwicklungen im Bereich der Wireless Sensor Networks (z.B. IP für Sensornetze [46, 48] oder IPv6 Routing Protocol for LLNs (RPL) [166]). Daher erscheint es sinnvoll neben iSense sowohl TinyOS als auch Contiki zu verwenden. Der Test auf Interoperabilität anhand einer simplen Kommunikations-Anwendung zeigt, dass ein OS-heterogenes Netz derzeit nicht möglich ist. Somit ist für jedes OS ein eigenes Netz notwendig, welche jedoch via Gateways verbunden werden können.

2.2 Routing

In der Literatur ist eine Vielzahl unterschiedlicher Routing-Ansätze für WSNs zu finden. Einen Überblick über diese Vielzahl wird beispielsweise in [3,4] und [2] verschafft. Im Rahmen des Berichts ist es nicht möglich, auf jeden dieser Ansätze einzugehen. Stattdessen wird sich auf eine Vorauswahl populärer Protokolle beschränkt und deren Einsatzgebiete, sowie Vor- und Nachteile diskutiert. Dazu werden zunächst die wesentlichen Herausforderungen für ein effizientes Routing-Verfahren in WSNs aufgeführt und explizit auf Unterschiede zum Routing in

mobilen ad hoc Netzwerken eingegangen. Abschließend erfolgt eine Auswahl der Protokolle, die im Testbed eingesetzt werden.

2.2.1 Herausforderungen

Die grundlegenden Herausforderungen für ein effizientes Routing in WSNs lassen sich in Herausforderungen, die in ähnlichem Maße allgemein für drahtlose multi-hop ad hoc Netzwerke gelten, und in WSN-spezifischen Herausforderungen einteilen. Diese werden im Folgenden vorgestellt.

Allgemeine Herausforderungen drahtloser ad hoc Netzwerke

Topologieänderung Das Bewahren der Konnektivität und die Aufrechterhaltung von Verbindungen unter Veränderung der Netzwerktopologie stellen wesentliche Herausforderungen für das Routing in mobilen ad hoc Netzwerken dar. Obwohl in einer Vielzahl von WSNs Sensorknoten statisch platziert sind, ist die Topologie auch in diesen Netzwerken aufgrund von Schwankungen der Verbindungsqualität in der Funkübertragung häufig dynamisch. Zusätzlich tragen Fehlfunktionen und Ausfälle von Sensorknoten zu dieser Dynamik bei [3].

Selbstorganisation Da in drahtlosen ad hoc Netzwerken und in der Regel auch in WSNs keine zentrale Infrastruktur vorhanden ist, ist die Autonomie der Netzwerkteilnehmer eine fundamentale Anforderung an das Routing-Protokoll [4]. Das Netz muss selbstorganisierend sein und das Hinzukommen bzw. den Ausfall von Knoten bewerkstelligen können. Dies gilt insbesondere für WSNs, in denen Sensorknoten nicht manuell installiert werden, sondern auf zufällige Weise angeordnet sind, wie beispielsweise bei der Ausbringung von Sensorknoten aus einem Flugzeug.

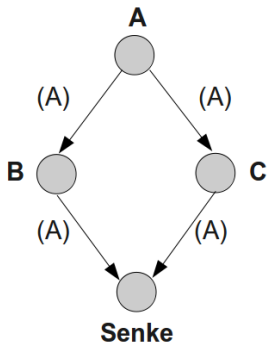
Skalierbarkeit Das eingesetzte Routing-Protokoll muss skalierbar in der Anzahl der Knoten des Netzwerks sein. Die Skalierbarkeit ist insbesondere in WSNs eine grundlegende Herausforderung, in denen die Anzahl der Knoten und die Konnektivität um ein Vielfaches höher ist als in einem typischen drahtlosen ad hoc Netzwerk [4].

Unterschiede zu mobilen ad hoc Netzwerken

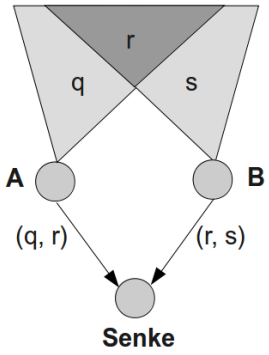
Neben den oben genannten allgemeinen Herausforderungen bestehen einige grundlegende Unterschiede zwischen drahtlosen Sensornetzen und (mobilen) multi-hop ad hoc Netzwerken. Diese besitzen Auswirkungen auf das Design effizienter Routing-Protokolle und sind bei der Adaption von Routing-Ansätzen aus dem Bereich mobiler ad hoc Netzwerke zu beachten. Die wesentlichen Unterschiede werden im Folgenden aufgeführt.

Datenverkehr Bei dem Datenverkehr in WSNs handelt es sich überwiegend um *Convergecasts (Concasts)*. Bei einem Concast senden mehrere Datenquellen (Sensorknoten) an eine Datensenke (Basisstation) [2, 4]. Neben Concasts kommen in die entgegengesetzte Richtung – von der Basisstation an alle Sensorknoten oder an Knoten in einem bestimmten Gebiet – *Broadcasts* oder *Geocasts* zum Einsatz. Dagegen ist der Austausch von Daten zwischen einzelnen Sensorknoten (P2P) in WSNs selten.

Redundanz Neben der Redundanz, die auch in mobilen ad hoc Netzwerken durch Multi-path-Übertragung auftreten kann und als Implosion bezeichnet wird (vgl. Abbildung 2.3(a)), besitzt der Datenverkehr in WSNs selbst potentiell erhebliche Redundanz. Dies ergibt sich beispielsweise dadurch, dass ein Ereignis, das von einer Vielzahl benachbarter Sensorknoten in einem geographischen Bereich detektiert wird, das Generieren gleicher oder identischer Daten in diesen Knoten auslöst [2, 4]. Diese Problematik wird als Überlappungsproblem bezeichnet und ist in Abbildung 2.3(b) skizziert.



(a) Das Implosionsproblem: Der Sensorknoten A flutet seine Daten (A) an seine beiden Nachbarknoten B und C, die beide diese Daten an die Datensenke weiterleiten. Vorausgesetzt es treten keine Paketverluste auf, erhält die Senke so ein zweites, redundantes Datenpaket.



(b) Das Überlappungsproblem: Sensorknoten, dessen zu überwachende Regionen sich überlappen, generieren evtl. die identischen Sensordaten. Hier beobachten die Knoten A und B in der überlappenden Region das Phänomen r, das von beiden Knoten an die Senke übermittelt wird.

Abbildung 2.3: Zwei grundlegende Herausforderungen für Routing-Protokolle in WSNs: Die Übertragung redundanter Daten führt zu einem erhöhten Energiekonsum des WSNs und einer Belastung der verfügbaren Bandbreite (Abbildung aus [73]).

Ressourcen-Limitierung Wie bereits erwähnt sind die Ressourcen der Sensorknoten stark limitiert. In der Regel stehen ausschließlich Energiequellen mit geringer Kapazität zur Verfügung, woraus sich eine signifikante Einschränkung der für die Übertragung nutzbaren Energie ergibt [2, 3], was die Beschränkung der Bandbreite nach sich zieht.

Fehleranfälligkeit Sensorknoten sind anfällig für Ausfälle durch Fehlfunktionen oder durch Erschöpfung der verfügbaren Energie [3]. Je nach konkretem Einsatzszenario besteht zudem die Gefahr von physischer Beschädigung oder Entwendung des Sensorknotens.

Positionsbestimmung Die Positionsbestimmung mittels Global Positioning System (GPS) ist aus mehreren Gründen in WSNs häufig nicht praktikabel. Je nach Szenario und Dichte des Sensornetzes ist es möglich, dass entweder kein Empfang von GPS-Informationen gegeben ist oder die Genauigkeit der Positionsinformationen über GPS nicht ausreicht [24]. In Routing-Protokollen und Anwendungen, in denen Positionsinformationen benötigt werden, können diese stattdessen z.B. durch Triangulationsverfahren erhalten werden [4] (vgl. auch Kapitel 7).

Anwendungsspezifische Kriterien mit Einflüssen auf das Protokolldesign

Die Ressourcenlimitierung der Hardware-Komponenten in WSNs erfordert ein effizientes Routing. Um dies zu erreichen, werden für WSNs entwickelte Routing-Protokolle häufig stark auf konkrete Anwendungen zugeschnitten. Dabei besteht ein Trade-off zwischen der Generalität und der Optimalität eines Routing-Protokolls und dem vom Protokoll verursachten Kontroll-Overhead. Fundamentale spezifische Kriterien konkreter Anwendungstypen, die in unterschiedlichem Maß von diversen Protokollen berücksichtigt werden, sind nachfolgend aufgelistet.

Data Delivery Modell Der Datentransport von der Quelle zur Senke in verschiedenen Anwendungen drahtloser Sensornetze lässt sich mittels drei Data Delivery Modellen beschreiben [2, 4]. Im *kontinuierlichen* Modell überträgt ein Sensorknoten gewonnene Daten periodisch an die Datensenke, während dieser Datentransport im *ereignisbasierten* Modell durch die Beobachtungen eines festgelegten Phänomens ausgelöst wird. Im *anfragebasierten* Modell verbleibt ein Sensorknoten bezüglich seiner Netzwerkaktivität solange in einem passiven Zustand, bis er von einer Basisstation zur Übertragung seiner aufgezeichneten Daten veranlasst wird. Zudem existieren hybride Anwendungen, die Kombinationen dieser Modelle verwenden.

Homogenität In Abhängigkeit der Anwendung ist die Menge der im Sensornetz eingesetzten Knoten homogen oder heterogen [2, 4]. Ein Grund für eine heterogene Zusammensetzung des Sensornetzes ist es beispielsweise, unterschiedliche Knoten für spezielle Aufgaben einzusetzen. Falls in einem heterogenen Netzwerk leistungsfähigere Knoten im Bezug auf Energie- und Rechenkapazität existieren, ist es von erheblichem Vorteil, wenn ein Routing-Protokoll diese Fähigkeiten berücksichtigt.

In-network-processing Die Redundanz bzw. die Ähnlichkeit der gewonnenen Sensordaten bietet die Möglichkeit, Sensordaten verschiedener Knoten während des Transports zum Empfänger in Zwischenknoten zu aggre-

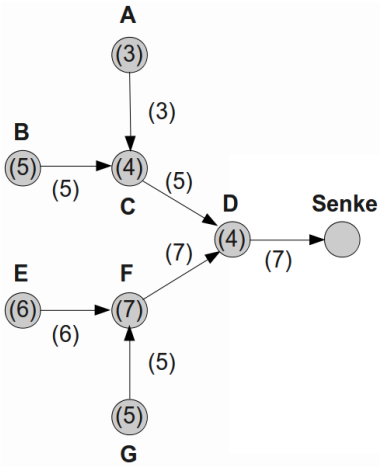


Abbildung 2.4: In-network-processing: Aggregation von Sensordaten: Die in jedem Knoten gewonnenen Daten (dargestellt durch die eingeklammerten Werte in den Knoten) werden zur Datensenke übermittelt und dabei aggregiert. Zur Aggregation dient hier die max-Funktion, d.h. es wird stets der maximale Wert weitergeleitet.

gieren. Dies ist in Abbildung 2.4 dargestellt und wird von einer Vielzahl der Anwendungen gefordert [2, 4]. Daher ist es erstrebenswert, dass ein Routing-Protokoll die Aggregation von Daten zu einem möglichst frühen Zeitpunkt während der Propagierung unterstützt. Allerdings stellt die Aggregation von Daten eine Herausforderung für die Ende-zu-Ende Sicherheit dar, weil Zwischenknoten Zugriff auf die Nutzdaten der weiterzuleitenden Paket besitzen [4].

Fehlertoleranz Das Spektrum der Anwendungen von drahtlosen Sensornetzen unterscheidet sich in dem Grad der Fehlertoleranz [3, 4]. Insbesondere in sicherheitsrelevanten, ereignisbasierten Applikationen wird ein hohes Maß an Zuverlässigkeit beansprucht, während auf der anderen Seite Anwendungen tolerant gegenüber einer gewissen Fehleranfälligkeit sind, in denen eine Vielzahl von Sensorknoten periodisch und mit hoher Frequenz Daten generieren und versenden.

2.2.2 Überblick

Dieser Unterabschnitt verschafft einen Überblick über eine Vorauswahl von Routing-Protokollen, die als Kandidaten für den Einsatz im Testbed in Frage kommen. Dabei wird auf Stärken und Schwächen, evtl. Anforderungen und Einsatzgebiete eingegangen.

Sensor Protocols for Information via Negotiation (SPIN)

Das Routing-Protokoll SPIN wurde im Jahr 2000 von Heinzelman et al. [73] vorgeschlagen. Es ist das erste *datenorientierte* (data-centric) Protokoll und für den Einsatz in ereignisbasierten WSNs vorgesehen.

Das Konzept von SPIN ist ein 3-Phasen-Handshake mit sogenannten Metadaten-Paketen, die die eigentlichen Daten-Pakete durch eine injektive Abbildung eindeutig beschreiben, jedoch eine signifikant geringere Größe aufweisen. Da das Format der Metadaten anwendungsspezifisch ist, wird es in SPIN nicht näher spezifiziert. Liegen einem Knoten im Netzwerk neue Daten vor, teilt er dies in einem Advertisement (ADV) durch ein solches Metadaten-Paket per Broadcast seinen Nachbarknoten mit. Bei Bedarf fordern die Nachbarn daraufhin durch ein Request (REQ) Paket die in dem ADV angebotenen Daten an, die dann im nächsten Schritt von der Datenquelle übermittelt werden. Alle Knoten, die mittels diesem 3-Phasen-Handshake (ADV, REQ, DATA) neue Daten erhalten haben, haben nun die Möglichkeit, weitere Daten mit diesen zu aggregieren und erneut durch ein ADV anzubieten, um auf diese Weise Daten im Netzwerk zu propagieren.

Der 3-Phasen-Handshake, in dem Daten explizit angefordert werden müssen, stellt sowohl für das Implosionsproblem als auch für das Überlappungsproblem eine Lösung dar. Daher trägt er unter der Voraussetzung, dass die Größe der Metadaten-Pakete signifikant geringer ist als die Größe der Datenpakete, zur Reduktion der benötigten Übertragungsenergie bei.

SPIN benötigt keine Topologieinformationen und kommt ohne zentrale Instanz aus. Stattdessen wird angenommen, dass jeder Knoten eine potenzielle Basisstation ist. Eine Einschränkung ist jedoch, dass SPIN voraussetzt, dass alle Zwischenknoten auf dem Weg von der Quelle zur Senke an den angebotenen Daten interessiert sind. Andernfalls können die Daten nicht bis zur Senke propagiert werden. Aus diesem Grund kann SPIN keine Garantie für die Datenübertragung gewährleisten.

Energy-Aware Routing (EAR)

Das strikte Verwenden von Pfaden, die entsprechend der eingesetzten Routing-Metrik optimal sind, kann ein Ungleichgewicht des Energieverbrauchs der Sensorknoten implizieren, da diese Pfade i.A. stärker beansprucht werden. In WSNs mit einer ausgezeichneten Datensenke sind insbesondere Sensorknoten in Nähe der Senke von diesem Effekt betroffen. Nach einem vorzeitigen Ausfall dieser Knoten ist die Senke nicht mehr erreichbar. Bei einer derartigen Partitionierung des Netzwerks ist daher die maximale Lebensdauer des gesamten WSNs durch die Lebensdauer dieser Knoten beschränkt.

Das Ziel des reaktiven Routing-Protokolls EAR [150] ist das Verhindern einer solchen Partitionierung durch einen gleichmäßigeren Energieverbrauch im Netzwerk. Um dies zu erreichen, wird eine randomisierte Auswahl einer Route aus

einer Menge suboptimaler Pfade getroffen. Dadurch wird das strikte Nutzen der optimalen Route vermieden.

EAR ist für anfragebasierte WSNs konzipiert. Das Aufbauen und Aktualisieren der Routen in EAR erfolgt durch *Backward-learning* über ein lokalisiertes Flooding, das von der Basisstation (Senke) initiiert wird. Dies reduziert den Overhead des netzwerkweiten Flutens und vermindert das Implosionsproblem, erfordert jedoch Positionsinformationen in den einzelnen Sensorknoten. Es wird eine klassenbasierte Adressierung mittels eines (Ort, Typ)-Tupels eingesetzt. Daher ist es möglich, dass eine Anfrage der Basisstation, die ein an ein bestimmtes Tupel adressiertes Interest-Paket versendet, von mehrere Knoten beantwortet wird.

Low-Energy Adaptive Clustering Hierarchy (LEACH)

Das hierarchische Routing-Protokoll LEACH [72] beruht wie EAR auf der Beobachtung, dass der ungleichmäßige Energieverbrauch die Lebensdauer eines WSNs einschränken kann. Das Protokoll ist für homogene WSNs ausgelegt, in denen es eine fixe Basisstation gibt und alle Sensorknoten periodisch Daten an diese Senke übertragen (kontinuierliches Data Delivery Modell). LEACH schlägt zur Lösung des ungleichen Energieverbrauchs eine Hierarchie mit Clustern vor, deren Cluster-heads in einem rundenbasierten und randomisierten Verfahren adaptiv nach vorhandener Restenergie ausgetauscht werden.

Die Kommunikation zwischen der Basisstation und den Cluster-heads erfolgt per Single-hop-Übertragung. Wegen der zufälligen Auswahl der Cluster-heads impliziert dies, dass sich beim Einsatz von LEACH alle Knoten im WSN in potentieller Sendereichweite zur Basisstation befinden müssen. Des Weiteren setzt LEACH die dynamische Adaption der Sendereichweite voraus. Andernfalls würde das Verfahren nicht zu einer Reduktion der Übertragungsenergie beitragen können.

Der Medienzugriff innerhalb eines Clusters erfolgt per Time Division Multiple Access (TDMA). Das Scheduling wird dabei durch den Cluster-head vorgenommen. Um den gemeinsamen Medienzugriff zwischen einzelnen Clustern und der Basisstation zu gewährleisten, wird dagegen Code Division Multiple Access (CDMA) eingesetzt. Aufgrund der Organisation in Clustern bietet LEACH eine einfach handhabbare Möglichkeit für In-network-processing.

Energy-Aware Data-centric routing (EAD)

Bei EAD [24] handelt es sich wie auch bei LEACH um ein datenorientiertes Protokoll, das für WSNs mit kontinuierlichem Data Delivery Modell konzipiert ist und In-network-processing unterstützt. Im Gegensatz zu LEACH wird ein Baum mit der Basisstation als Wurzel erzeugt, der als virtuelles Backbone-Netz bezeichnet wird und nicht in der Tiefe beschränkt ist. Somit eignet sich

EAD auch für WSNs, in denen sich Sensorknoten nicht notwendigerweise in Sendereichweite der Basisstation befinden.

Analog zu den Sensorknoten eines Clusters in LEACH können die Blatt-Sensorknoten in EAD pausieren, wenn sie keine Daten zu übertragen haben. Lediglich die Knoten des virtuellen Backbones müssen dauerhaft aktiv sein. Auf diese Weise kann Energie eingespart und die Lebensdauer des Sensornetzes verlängert werden.

Das virtuelle Backbone-Netz muss flächendeckend sein, um die Konnektivität jedes Knotens zu gewährleisten. Gleichzeitig ist es erstrebenswert, die Anzahl der Backbone-Knoten minimal zu wählen. Die Konstruktion eines Spannbaums mit maximaler Anzahl von Blättern ist jedoch ein NP-schweres Problem, d.h. es ist kein polynomieller Algorithmus zur Lösung des Problems bekannt. Daher verwendet EAD eine Heuristik, welche die verbleibenden Energiereserven berücksichtigt. Der Aufbau des Backbones, der von der Senke initiiert wird, erfolgt durch Broadcasting von Kontrollnachrichten und wird rundenbasiert ausgeführt. Dadurch wird die gleichmäßige Verteilung der Energiebelastung erreicht. Topologieänderungen können jedoch erst mit der nächsten Runde erkannt werden.

Collection Tree Protocol (CTP)

Der Anspruch des baumbasierten CTP [66] ist neben einer effizienten und zuverlässigen Datenübertragung die Robustheit des Algorithmus. Ohne manuelle Konfiguration und Tuning von Parametern wird die Einsetzbarkeit von CTP in einem breiten Feld von verschiedenen WSNs angestrebt, die auf dem kontinuierlichen Data Delivery Modell basieren.

Die beiden Schlüsseltechniken von CTP sind *Adaptive-beaconing* und *Datapath-validation*. Das Adaptive-beaconing wird als Neighbor-discovery-Mechanismus eingesetzt und dient zur Konstruktion eines Baums (pro Datensenke). Die Knoten der Bäume sind nach steigenden Kosten, die für den Transport der Daten benötigt werden, angeordnet. Als Kostenmetrik wird die Expected Transmission Count (ETX)-Metrik verwendet. Falls mehrere Senken existieren, werden die Daten automatisch an die Senke mit minimalen Kosten übermittelt.

Das Intervall, mit dem die Beacons gesendet werden, ist in konventionellen Protokollen, wie z.B. EAD, fix. Mit der Größe des Intervalls wird hier der Trade-off zwischen dem Routing-Overhead und der Latenz bei Reaktion auf Topologieänderungen festgelegt. Im Gegensatz dazu verwendet CTP den *Trickle-Algorithmus* [101] (siehe Abschnitt 2.3.2) zur adaptiven Anpassung des Beacon-Intervalls in Abhängigkeit der Topologiedynamik.

Datapath validation bezeichnet die Technik, den Datenverkehr zu nutzen, um Topologieinformationen zu gewinnen und Routing-Loops frühzeitig zu entdecken. Dies erfordert, dass jedes Datenpaket die Kosten enthält, die dem Absender zugeteilt sind. Erhält ein Knoten ein Paket mit geringeren Kosten als die

eigenen, ist dies ein Hinweis auf eine Inkonsistenz der Routing-Topologie. Ein solches Ereignis löst dann umgehend einen Versuch zur lokalen Reparatur der Inkonsistenz durch Versenden von Beacons aus.

IPv6 Routing Protocol for LLNs (RPL)

Das RPL [157] ist ein Distance Vector Routing Protokoll, das große Ähnlichkeit zu CTP aufweist. Derzeit wird es von der IETF Routing Over Low power and Lossy networks (ROLL) Working Group standardisiert. RPL wurde entworfen, um den speziellen Anforderungen von Low power and Lossy Networks (LLNs) gerecht zu werden. Dies sind insbesondere geringe Datenraten und Rechenleistung, strikte Energieeinschränkungen und hohe Paketverlustwahrscheinlichkeiten. RPL basiert auf dem zukunftsweisenden Internet Protocol Version 6 (IPv6). Des Weiteren sind in der Spezifikation von RPL Sicherheitsmechanismen vorgesehen, deren Implementierung optional ist.

Das Kernkonstrukt von RPL ist der zielorientierte gerichtete azyklische Graph (Destination Oriented Directed Acyclic Graph (DODAG)) mit je einer Basisstation als Wurzel. In RPL werden diese Basisstationen als LLN Border Router (LBR) bezeichnet, was den vorgesehenen Anschluss einer Basisstation an bestehende Netzwerke (z.B. Internet) betont, der insbesondere durch Nutzung von IPv6 ermöglicht wird. Nach einer gewissen Einschwingphase ist jeder Knoten an den DODAG angebunden und besitzt eine Menge von Elternknoten und folglich eine Menge alternativer Pfade zur Basisstation, von denen er einen Pfad präferiert. Dadurch ist Redundanz geschaffen, so dass bei Ausfall eines Elternknotens ein Ersatzknoten verfügbar ist. Sollte keiner der Elternknoten erreichbar sein, wird der Pfad über Geschwisterknoten im DODAG beansprucht.

Analog zu CTP erfolgt der Aufbau und die Instandhaltung des DODAGs durch den Broadcast von Kontrollnachrichten (DODAG Information Objects (DIOs)) an alle benachbarten Knoten, ausgehend von der Basisstation. Beim globalen Aktualisieren des DODAGs werden Sequenznummern eingesetzt, um verschiedene Versionen der DODAGs unterscheiden zu können. Ferner wird der Broadcast von DIOs wie auch in CTP adaptiv durch den Trickle-Algorithmus kontrolliert. DIOs enthalten unter anderem den Rang des Absenders, der anhand einer Zielfunktion (z.B. Minimierung von ETX) ermittelt wird. Auch in RPL ist neben der globalen Reparatur durch das Aktualisieren des DODAGs ein lokales Reparieren möglich.

Im Unterschied zu CTP und anderen baumbasierten Protokollen wird in RPL neben der Basisstation-zu-Sensorknoten Kommunikation (bzw. in umgekehrter Richtung) auch die P2P Kommunikation zwischen einzelnen Sensorknoten unterstützt.

Protokoll:	**SPIN**	**EAR**	**LEACH**	**EAD**	**RPL**	**CTP**
Qualität der Routing-Pfade	+	+	•	+	+	+
Kontroll-Overhead	–	•	•	•	+	+
Berücksichtigung der Restenergie	•	+	+	++	–	–
Implosionsproblem (Abbildung 2.3(a))	+	+	+	+	+	+
Überlappungsproblem (Abbildung 2.3(b))	+	•	•	•	•	•
Reaktion auf Topologieänderungen	++	+	•	•	++	++
Skalierbarkeit	–	+	+	•	+	+
In-network-processing	+	+	+	+	+	+
Data Delivery Modell (e ereignis-, a anfragebasiert, k kontin.)	e	a	k	k	k	k

Tabelle 2.9: Gegenüberstellung der vorgestellten Routing-Protokolle (+ symbolisiert ein positives, • ein neutrales und – ein negatives Ergebnis bezüglich des entsprechenden Kriteriums).

2.2.3 Fazit

In diesem Abschnitt wurde zunächst auf die Herausforderungen für die Effizienz von Routing-Protokollen beim Einsatz in WSNs eingegangen. Danach erfolgte ein Überblick über eine Vorauswahl diverser Protokolle. Im Folgenden wird nun eine Bewertung dieser potentiellen Kandidaten vorgenommen und eine Auswahl für das Testbed getroffen.

SPIN ist für WSNs mit geringer Fehlertoleranz ungeeignet, da die Übermittlung der Daten nicht garantiert werden kann. Dadurch impliziert das Protokoll im Vergleich zu EAR, LEACH und EAD einen höheren Routing-Overhead. Die drei zuletzt genannten Protokolle besitzen den Vorteil, dass sie die beschränkten Energieressourcen der Sensorknoten explizit berücksichtigen. Ein gravierender Nachteil des standortbezogenen (location-based) EAR ist jedoch, dass Positionsinformationen aller Knoten benötigt werden, was die Betrachtung von EAR im Rahmen dieses Projekts ausschließt. LEACH besitzt den Nachteil, dass für die Inter-cluster-Kommunikation CDMA erforderlich ist, das von IEEE 802.15.4-konformer Hardware aber nicht unterstützt wird. Zudem ist die Fläche, auf der sich das Sensornetz erstreckt, durch die Anforderung einer vollvermaschten Topologie eingeschränkt. Daher ist LEACH in der Praxis nur bedingt einsetzbar.

Sowohl EAD als auch RPL und CTP erstellen Routing-Topologien, die die einfache und effiziente Aggregation von Daten ermöglichen. Die beiden auf Trickle basierenden Protokolle RPL und CTP besitzen allerdings gegenüber EAD den Vorteil der adaptiven Anpassung der Routing-Nachrichten zur Reduktion des Overheads. In Tabelle 2.9 sind zusammenfassend die Vor- und Nachteile aller Routing-Protokolle bezüglich verschiedener Kriterien gegenübergestellt.

Für die in Betracht gezogenen Betriebssysteme ist nach jetzigem Kenntnisstand (08/2010) keine Implementierung der Protokolle SPIN, EAR, LEACH oder EAD erhältlich. Dagegen sind Implementierungen des werdenden Standards RPL sowohl für TinyOS, als auch für Contiki verfügbar. Aus diesem Grund

– und nicht zuletzt wegen der oben aufgeführten Vorteile (IPv6 und Sicherheitsmechanismen) – wurde entschieden, RPL als Routing-Protokoll im Testbed einzusetzen. Darüber hinaus ist vorgesehen, als weiteres Routing-Protokoll ein baumbasiertes Verfahren zu untersuchen. Dazu bietet es sich an, in TinyOS auf das Default-Protokoll CTP und im iSense-Betriebssystem auf ein dort vorhandenes baumbasiertes Protokoll zurückzugreifen.

2.3 Automatische Updates und Fernwartbarkeit

Aufgrund der hohen Anzahl von Sensorknoten ist das manuelle Einspielen von Updates in das Sensornetz zumeist mit nicht handhabbarem Aufwand verbunden. Zudem existieren Anwendungen, in denen Sensorknoten physisch nicht erreichbar sind, sondern ausschließlich per Funkverbindung auf diese zugegriffen werden kann. Dennoch sind Programm-Updates zur Fehlerbehebung, zur (Re-)Konfiguration von Parametern oder zur Veränderung der Aufgaben und Funktionalitäten von Sensorknoten für den Langzeitbetrieb von WSNs – aber auch für den Betrieb einer Testumgebung – unerlässlich. Daher stellt die drahtlose und automatische Durchführung derartiger Updates bzw. das drahtlose Umprogrammieren (Over-The-Air Programming (OTAP)), ausgehend von einer zentralen Basisstation, für die Fernwartbarkeit von WSNs ein entscheidendes Kriterium dar.

Im Folgenden werden zunächst die Kernanforderungen beschrieben, die an Reprogramming-Protokolle zur Realisierung von OTAP gestellt werden. Anschließend wird ein Überblick über implementierte Ansätze gegeben, die derzeit (08/2010) von den Betriebssystemen angeboten werden. Abgeschlossen wird dieser Abschnitt mit einem Fazit und der Auswahl der Reprogramming-Protokolle für den Einsatz im Testbed.

2.3.1 Anforderungen

Eine inhärente Anforderung an ein Reprogramming-Protokoll ist die Zuverlässigkeit. Ein solches Protokoll muss sicherstellen, dass das propagierte Update vollständig und fehlerfrei empfangen wurde bevor das Update ausgeführt wird, um Fehlfunktionen von Knoten zu vermeiden. Zudem muss der zuverlässige Transport des Updates auch in Umgebungen mit hohen Paketverlustwahrscheinlichkeiten garantiert werden können.

Weitere Anforderungen stellen Skalierbarkeit und Robustheit dar, sowie eine geringe Dauer des Update-Prozesses, um die Operation des Sensornetzes durch den Prozess nur möglichst kurzzeitig zu unterbrechen und die Knoten zeitnah zu aktualisieren.

Darüber hinaus ist eine Berücksichtigung der beschränkten Energieressourcen in WSNs eine eminent wichtige Anforderung. Dazu ist zum einen eine geringe Speicheranforderung des Reprogramming-Protokolls bzw. eine geringe Anzahl von

Speicherzugriffen dienlich und zum anderen die Reduktion der bei der Verbreitung des Updates erforderlichen Übertragung. Letzteres impliziert die Minimierung von benötigten Kontrollnachrichten und die Vermeidung von Redundanz, sowie die Kompaktheit des Updates. Die meisten Reprogramming-Protokolle verwenden kompilierte Programm-Images zur Aktualisierung der Sensorknoten [172]. Der Overhead dieser Methode ist wegen der Größe dieser Images jedoch relativ hoch, wenn durch das Update nur geringe Änderungen an dem Programm-Code vorgenommen werden. In einem solchen Fall sind inkrementelle Updates geeigneter, die beispielsweise in dem Incremental Network Programming Verfahren [85] oder dem Ansatz von Reijers et al. [137] verwendet werden.

Des Weiteren ist es wünschenswert, dass ein Reprogramming-Protokoll multihop-fähig ist und das gezielte Umprogrammieren einer Teilmenge der Sensorknoten eines WSNs unterstützt, was in heterogenen WSNs zwingend erforderlich sein kann. Aber auch in homogenen Sensornetzen kann das Umprogrammieren einer Menge selektierter Knoten nützlich sein, um diesen spezielle Aufgaben zuzuteilen.

2.3.2 Überblick

Ein umfassender Überblick über diverse Reprogramming-Ansätze ist in [25] und in [172] zu finden. Im Rahmen dieses Berichts ist es nicht möglich, auf jedes der in beiden Überblicken genannten Protokolle einzugehen. Stattdessen werden im Folgenden die Konzepte dreier populärer und in der Praxis verbreiteter Protokolle vorgestellt, von denen Implementierungen für die in Betracht gezogenen Betriebssysteme existieren.

Jedes dieser Protokolle unterstützt die Multi-hop-Propagierung der Updates. Bei den Updates handelt es sich in allen Ansätzen – mit Ausnahme von Trickle – stets um ein komplettes, kompiliertes Programm-Image, das zur Übertragung in eine Vielzahl von Paketen fragmentiert wird. Keines der hier vorgestellten Protokolle bietet die Option, inkrementelle Updates durchzuführen oder die Aktualisierungen auf eine Teilmenge der Knoten zu beschränken.

Multihop Over-the-Air-Programming (MOAP)

MOAP [155] wurde in TinyOS implementiert und zielt speziell auf die MICA2-Plattform ab. Das Protokoll basiert auf einem Publish-Subscribe-Ansatz, bei dem jeder Knoten (begonnen bei der Basisstation) in einem Publish-Paket die Version seines aktuellen Programm-Codes anbietet. Knoten, auf denen zum Zeitpunkt des Empfangs eines solchen Pakets eine geringere Version vorhanden ist, fordern den Absender durch ein Subscribe-Paket dazu auf, mit dem Transfer des Updates zu beginnen. Dabei wird überprüft, ob die Linkqualität zwischen dem Anbieter und dem Interessenten zur Durchführung des Updates ausreicht.

Der Publish-Subscribe-Ansatz dient zur Reduktion redundanter Übertragungen. Zur Fehlerbehandlung wird zusätzlich ein Sliding-Window-Mechanismus mit Negative ACKs (NACKs) eingesetzt. Um den Zugriff auf den Electrically Erasable Programmable Read-Only Memory (EEPROM) zu minimieren, in dem empfangene Update-Fragmente gespeichert werden, wird im RAM ein Bit-Vektor zur Verwaltung unvollständiger Fragmente angelegt. Erst nachdem ein Knoten das vollständige Programm-Update empfangen hat, bietet er dieses mittels Publish-Paket seiner Nachbarschaft an (Store-and-forward-Prinzip). Unter Umständen verursacht diese Single-hop-Verbreitung jedoch beim Update-Prozess hohe Latenzen.

Trickle

Trickle [101] ist ein skalierbarer und selbst-regulierender Algorithmus, der auf dem dynamischen Austausch von Metadaten basiert und sich zur schnellen Verbreitung von Programm-Code im Sensornetz eignet. Der Algorithmus zeichnet sich durch geringen Verwaltungsoverhead und gute Skalierbarkeit aus, die kein a priori Wissen über die Netzwerkdichte erfordert.

Ähnlich wie MOAP wird in Trickle zur Verbreitung der Updates ein periodisches Broadcasten der Code-Versionen in einem Metapaket genutzt. Allerdings handelt es sich hier um ein verbessertes Verfahren zur Reduktion redundanter Übertragungen und zur verbesserten Skalierbarkeit. Die Idee dieses Verfahrens, das *Polite Gossip* genannt wird, ist, dass jeder Knoten, der Metadaten mit der identischen Versionsnummer empfängt, das Senden eigener Metadaten unterdrückt. Werden dagegen Metadaten mit veralteten Versionsnummern empfangen, stellt dies eine Aufforderung zum Anbieten der aktuellen Version dar.

Als Maßnahme gegen Link-Asymmetrien, Multi-path-Effekte, Verbindungsabbrüche, Knotenfehler, etc. ist es erforderlich, dass der Update-Prozess kontinuierlich vollzogen wird. Auf diese Weise können auch Knoten, die erst nach Durchführung eines Updates dem Netzwerk beitreten, erreicht werden. Die Kontinuität des Broadcasts der Metapakete erzeugt einen hohen Overhead hinsichtlich der für das Senden erforderlichen Energie. Zur Vermeidung dieses Overheads dient die zweite Idee des Trickle-Algorithmus, die das adaptive und lokale (pro Knoten) Regulieren der Sendeperiode der Metadaten vorsieht. Werden beispielsweise in einem WSN mit stabiler Topologie, in dem alle Knoten den aktuellen Programm-Code besitzen, zeitweise keine Updates eingespielt, reguliert sich die Sendeperiode auf ein Maximum, während diese dynamisch gesenkt wird, sobald Bedarf nach der Verbreitung von Updates entsteht.

In Trickle wird angenommen, dass sich das zu propagierende Update in einem einzigen Datenpaket übertragen lässt. Weil die typische minimale Größe eines Programm-Images jedoch im Bereich von mehreren kBps liegt, stellt diese Annahme eine massive Einschränkung für den Einsatz von Trickle dar. Auf der anderen Seite entfällt bei Trickle dadurch der Bedarf nach einer effizienten Speicherverwaltung.

Deluge

Deluge [81] ist das Standard-Reprogramming-Protokoll von TinyOS und basiert auf dem Trickle-Algorithmus. Es erweitert Trickle um die Fähigkeit, große Programm-Images (fragmentiert in mehrere Pakete) für den Update-Prozess nutzen zu können. Gleichzeitig nutzt es das Polite Gossip von Trickle, sowie die adaptive Regulierung der Broadcast-Perioden. Deluge verwendet einen 3-Phasen-Handshake (ADV, REQ, DATA), um analog zu MOAP und Trickle auf diese Weise Redundanz zu reduzieren. Zudem dient der Handshake der Überprüfung auf Bidirektionalität des Links zwischen Anbieter des Updates und dessen Interessenten. Es ist nicht erforderlich, dass ein Knoten, der Fragmente eines Updates anbietet, dieses vollständig empfangen hat, wie es bei MOAP der Fall ist. Dadurch ist in Deluge eine schnellere Ausbreitung des Updates möglich. Die ADV- und REQ-Pakete enthalten einen Bit-Vektor, um die bereits vorliegenden Pakete eines Anbieters bzw. die fehlenden Pakete eines Interessenten in komprimierter Weise darzustellen.

Wie auch in den beiden zuvor vorgestellten Protokollen ist in Deluge keine zentrale Instanz zur Koordination des Update-Prozesses erforderlich. Zudem biete Deluge, die Möglichkeit, mehrere Programm-Images auf einem Knoten zu speichern und über einen Bootloader zwischen diesen zu wechseln [80].

2.3.3 Fazit

Die vorgestellten, derzeit verfügbaren Implementierungen werden nicht der Anforderungen von OTAP-Systemen gerecht, gezielt eine Teilmenge von Knoten eines Sensornetzes aktualisieren zu können. Der Fokus dieses Projekts liegt jedoch nicht auf der Entwicklung und Implementierung von Reprogramming-Protkollen. Daher wird sich an dieser Stelle für eines der vorgestellten Protokolle und dessen Einsatz im Testbed entschieden.

Trickle und Deluge versprechen im Vergleich zu MOAP den Vorteil einer besseren Skalierbarkeit und eines geringeren Energiebedarfs aufgrund des Polite Gossips und der adaptive Regulierung der Sendeperiode. Darüber hinaus hebt Deluge die Einschränkung von Trickle auf, die sich daraus ergibt, dass in Trickle ausschließlich *ein* Paket für das Update genutzt werden kann. Aus diesem Grund wurde sich dazu entschieden, für das automatisierte Update der Testumgebung auf Deluge zurückzugreifen, von dem bereits eine Implementierung für TinyOS und Contiki existiert.

Das iSense-Betriebssystem unterstützt z.Z. lediglich ein Single-hop-OTAP, das die Grundfunktionalität für die Aktualisierung bereitstellt. Eine komplexe Multi-hop-Lösung, die im Gegensatz zu den aufgeführten Protokollen sogar die Umprogrammierung einer Teilmenge von Knoten ermöglicht, befindet sich in der Entwicklung [94].

2.4 Topologie-Management im Testbed

In diesem Abschnitt wird auf das Topologie-Management des Testbeds eingegangen. Dazu werden zunächst verschiedene alternative Methoden und deren charakteristische Eigenschaften diskutiert. Anschließend wird in einem Fazit eine Methode ausgewählt und die getroffene Wahl begründet.

2.4.1 Überblick

Eine Anforderung an das Testbed ist, dass unterschiedliche Topologien kontrolliert untersucht werden können. Diese reichen von einfachen *Stern-Topologien*, bei der jeder Sensorknoten über einen direkten Link mit einer zentralen Basisstation verbunden ist, über komplexere Topologien, die Multi-hop-Verbindungen erfordern, wie *Cluster-* und *Baum-Topologien*, bis hin zu einer *vermaschten Topologie*. Die in der Praxis angestrebten Anwendungen basieren meist auf dem letzteren Topologietyp. Die Mehrzahl der in Abschnitt 2.2 vorgestellten Routing-Protokolle bieten die Möglichkeit, in physisch teil- oder vollvermaschten Topologien logische Topologiestrukturen, wie Cluster- und Baum-Topologien, für den effizienten Datentransport herzustellen.

Zur Konstruktion eines Testbeds, das gezielt die oben genannten Topologien unterstützt, existieren drei alternative, grundlegende Methoden. Diese werden im Folgenden vorgestellt.

Physisches Topologie-Management

Bei der Methode des physischen Topologie-Managements sind die Sensorknoten so zu platzieren, dass sich die gewünschten Topologien aus den Konnektivitäten der Knoten ergeben, die von der Distanz zwischen einzelnen Knoten und deren Sendereichweite abhängt. Dieses Platzieren ist jedoch relativ aufwendig und impliziert, dass eine Modifikation der gewünschten Topologie den manuellen Umbau des Testbeds erforderlich macht.

Der Vorteil dieser Methode ist, dass sich das Verhalten von realen Anwendungen, denen die konstruierte Topologie zugrunde liegt, unverfälscht wiedergeben lässt. Allerdings stellt die Reproduzierbarkeit der Ergebnisse ein inhärentes Kernproblem dar. Die in der Praxis zu beobachtenden dynamischen Schwankungen in der Signalausbreitung führen unter anderem zu variierenden Sendereichweiten. Unter Umständen kann dies eine Veränderung der beabsichtigten Topologie zur Folge haben. Des Weiteren wird bei Sendereichweiten von etwa 100 m je nach beabsichtigter Topologie ein großes Areal für das Testbed benötigt. Dies erschwert zusätzlich die Instandhaltung des Sensornetzes, da fehlerhafte Knoten lokalisiert und erreicht werden müssen.

Regulierung der Sendeleistung

Die verfügbare Hardware erlaubt das Regulieren der Sendeleistung und damit einhergehend das Regulieren der möglichen Übertragungsdistanz. Durch ein Verringern der Sendeleistung ist es möglich, das Areal des Testbeds im Gegensatz zu dem physischen Topologie-Management signifikant zu verkleinern und so auch die Wartung des Netzwerks zu erleichtern. Der Einsatz geringerer Sendestärken beeinflusst jedoch massiv die physikalischen Effekte der Funkübertragung. Auf diese Weise wird auf der einen Seite das Verhalten realer Anwendungen verfälscht. Darüber hinaus wird auf der anderen Seite das inhärente Kernproblem der Reproduzierbarkeit im Vergleich zur vorherigen Methode noch verstärkt. Zudem wird die Beeinträchtigung durch potentiell existierende Interferenzen aus WLAN-Netzen intensiviert.

Virtual Links

Das im Rahmen des EU-Projekts Wireless Sensor Network Testbeds (WISEBED) [148] entwickelte Konzept der *Virtual Links* [15] stellt einen Ansatz zur Virtualisierung der physischen Topologien von Testbeds dar und bietet zwei Basisfunktionalitäten. Virtuelle Links können zum einen dazu eingesetzt werden, um physisch isolierte Testbeds zu verbinden (z.B. über das Internet). Insbesondere Testbeds mit heterogener Hardware können auf diese Weise zu *einem* virtuellen Netzwerk vereint werden. Zusätzlich besteht die Möglichkeit, simulierte Testbeds bestimmter Simulatoren an das reale Testbed anzugliedern. Zum anderen erlaubt das Konzept der Virtual Links die individuelle Kontrolle der gewünschten Topologie innerhalb eines Testbeds.

Das Testbed-interne Topologie-Management ermöglicht es auf der einen Seite, zwei Knoten, die sich gegenseitig nicht in direkter Kommunikationsreichweite befinden, über einen virtuellen Link zu verbinden. Dazu wird eine bestimmte Software-Komponente, *Virtual Radio* genannt, auf den Knoten installiert. Der virtuelle Link zwischen den beiden Knoten erfolgt dann über eine reale Verbindung zwischen den Knoten und einer zentralen Instanz (*Testbed Server*). Aus Sicht der Anwendung ist diese Umleitung transparent. Auf der anderen Seite besteht die Option, über das Virtual Radio gezielt selektierte Links eines Testbeds zu deaktivieren. Beide Funktionalitäten könne dynamisch während der Laufzeit des Testbeds erfolgen. Der Testbed Server bildet dabei die Benutzerschnittstelle und übernimmt das netzinterne Management der aktuellen Topologie durch Propagierung von Routing-Tabellen an das Virtual Radio-Interface jedes Knotens. Neben der dynamischen Kontrolle, die im Gegensatz zu beiden vorherigen Methoden zudem einfach zu handhaben ist, erleichtert das Konzept der Virtual Links zusätzlich die Reproduzierbarkeit von Testläufen und die Instandhaltung des Testbeds.

2.4.2 Fazit

Für das Topologie-Management im Testbed wurde sich dazu entschieden, auf den Ansatz der Virtual Links zurückzugreifen. Dieser Ansatz verspricht im Gegensatz zu den beiden anderen vorgestellten Methode eine bessere Reproduzierbarkeit und eine einfachere Möglichkeit, die verschiedenen gewünschten Topologien umzusetzen und bei Bedarf – sogar während der Laufzeit – kontrolliert zu modifizieren. Des Weiteren ermöglicht der Einsatz von virtuellen Links eine kompakte Indoor-Realisierung des Testbeds, die keine Verteilung der Hardware über mehrere Räumlichkeiten erfordert und die unmittelbare physische Erreichbarkeit der Sensorknoten sicherstellt.

2.5 Beschreibung des Konzepts für den Laboraufbau

In diesem Abschnitt erfolgt eine Zusammenfassung der für das Testbed ausgewählten Betriebssysteme und Protokolle sowie Hardware-Komponenten. Des Weiteren wird der Laboraufbau konkretisiert und auf weitere Hilfskomponenten eingegangen.

2.5.1 Überblick über die einzusetzenden Betriebssysteme und Protokolle

Dieser Unterabschnitt fasst die Entscheidungen bezüglich der Auswahl von OSs und der einzusetzenden Protokolle zusammen, die in den Abschnitten 2.1, 2.2 und 2.3 getroffen wurden.

Es werden die drei Betriebssysteme Contiki, iSense und TinyOS in Betracht gezogen. Auf MAC-Ebene wird das Standardprotokoll des jeweiligen Betriebssystems verwendet. Das Routing wird OS-übergreifend auf das IPv6-Protokoll RPL fokussiert. Zusätzlich sollen bei iSense und TinyOS jeweils die baumbasierten Standardprotokolle eingesetzt werden (siehe Abschnitt 2.3.3). Ferner ist beabsichtigt, die Ausführung von automatischen Updates bzw. das Umprogrammieren der Sensorknoten sowohl in TinyOS als auch in Contiki über Deluge bereitzustellen. Für iSense ist dagegen das derzeit verfügbare OTAP-Protokoll vorgesehen.

2.5.2 Überblick über den Laboraufbau

Die Betrachtung der drei ausgewählten Betriebssysteme macht es unter Berücksichtigung der derzeit nicht vorhandenen Interoperabilität (siehe Tabelle 2.8 in Abschnitt 2.1.3) erforderlich, das Testbed in drei Teilnetze zu unterteilen, denen je ein Betriebssystem zugeordnet ist (siehe Abbildung 2.5). Der parallele Betrieb dieser drei Sensornetze ist durch den Einsatz unterschiedlicher Funkkanäle sichergestellt. Dazu werden die Kanäle 11, 18 und 26 verwendet, die den maximalen Kanalabstand zueinander besitzen und so die räumliche Überlappung ohne gegenseitige Beeinträchtigung durch Interferenz ermöglichen. Jedes dieser Teilnetze verfügt über ein Gateway, das an eine zentrale Basisstation angebunden ist.

In den Teilnetzen, in denen TinyOS und Contiki verwendet werden, werden heterogene Hardware-Komponenten (TelosB und MicaZ) eingesetzt. Dagegen ist das iSense-Teilnetz homogen und basiert auf der iSense-Plattform Core Module CM10C. Jedes Teilnetz besteht jeweils aus 10 Sensorknoten. Je nach Anwendung kommen darin diverse Sensorboards zum Einsatz.

Das Topologie-Management wird durch das Konzept der virtuellen Links (Abschnitt 2.4.1) umgesetzt, das die dynamische Realisierung der gewünschten Topologien und den Aufbau des Testbeds in einem abgeschlossenen Raum gestat-

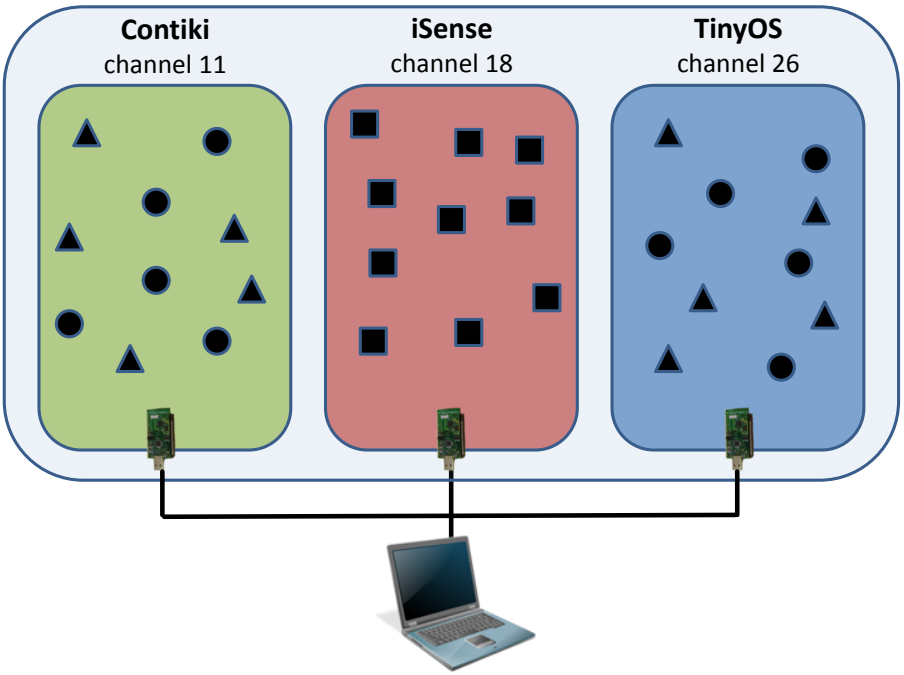


Abbildung 2.5: Schematisierte Darstellung des Laboraufbaus. Die Testumgebung unterteilt sich in drei parallele Netzwerke, die je einem Betriebssystem zugeordnet und an eine gemeinsame Basisstation angebunden sind.

tet, in dem sich alle eingesetzten Sensorknoten in gegenseitiger Kommunikationsreichweite befinden.

Als weitere Komponenten des Testbeds, die zur Realisierung von Angriffen und deren Analyse dienen, sind ein IEEE 802.15.4 Jammer und Paket-Sniffer vorgesehen. Beide werden im Folgenden kurz vorgestellt.

Paket-Sniffer Zusätzlich zu den bisher genannten Komponenten sollen Paket-Sniffer die Analyse der im Testbed ausgetauschten Pakete ermöglichen. Die Realisierung des Testbeds als vollvermaschtes Netzwerk, in dem sich unterschiedliche Topologien virtuell konstruieren lassen, ermöglicht es, *alle* gesendete Paket mit *einem* Sniffer pro verwendetem Funkkanal zu registrieren. Dabei ist angestrebt, dass Pakete zur Analyse eines Testlaufs und zum Debugging von Protokollen sowohl in einer Live-Ansicht dargestellt, als auch aufgezeichnet werden können.

Die Auswahl eines geeigneten, IEEE 802.15.4 kompatiblen Paket-Sniffers fiel auf den *AVR RZUSBSTICK* des Herstellers Atmel [12]. Das Betriebssystem Contiki bietet in seinem Sourcecode eine Applikation zur Integration des Sniffers in das weit verbreitete Protokollanalyse-Tool *Wireshark* [178] an, die auf das Gerät installiert werden muss. Für jedes der drei Netzwerke wird ein solcher Paket-Sniffer an die zentrale Basisstation angeschlossen.

IEEE 802.15.4 Jammer Als weitere Komponente des Testbeds wird ein IEEE 802.15.4 Jammer eingesetzt, mit dessen Hilfe gezielt Jamming-Angriffe durchgeführt werden können. Dabei soll die Kontrolle des Jammers ebenfalls per Remote-Verbindung möglich sein.

Zur Realisierung einer derartigen Jammer-Komponente empfiehlt sich der Einsatz von Software Defined Radio (SDR), das dem Benutzer erlaubt, das zum Jamming verwendete Signal frei zu definieren. Es wurde entschieden, als Software-Komponente für das SDR das freie Software Development Kit (SDK) *GNU Radio* [63] zu nutzen und für die Hardware-Komponente auf das *Universal Software Radio Peripheral* (USRP)-Board des Hersteller Ettus Research LLC [55] zurückzugreifen, mit dem sich sowohl die Signalstärke als auch die Frequenz und die Bandbreite des Störsignals konfigurieren lässt. Die Kombination aus GNU Radio und USRP bietet die benötigte Flexibilität, um verschiedenartige Angriffe durchführen zu können. Ein Überblick über unterschiedliche Jammer-Typen ist beispielsweise in [9] zu finden.

2.6 Zusammenfassung

In diesem Kapitel wurde das Konzept zum Aufbau des Testbeds vorgestellt. Dazu wurden zunächst Sensorknoten-Hardware sowie verschiedene Betriebssysteme für WSNs beschrieben. Letztere wurden weiterhin verglichen und evaluiert, was zu einer fundierten Entscheidung darüber geführt hat, welche Betriebssysteme innerhalb des Testbeds zum Einsatz kommen. Darauf folgte ein Überblick über verschiedene Routing-Protokolle für den Einsatz in WSNs. Dabei wurden jeweils Vor- und Nachteile diskutiert. Anschließend erfolgte im Fazit eine Bewertung und Auswahl der Protokolle für das Testbed. Danach wurden für die Fernwartbarkeit von WSN-Testbeds essentielle Ansätze für automatische Software-Updates verglichen. Bei dieser Übersicht lag der Fokus auf bereits implementierten Update-Protokollen und sie wurde ebenso durch eine Auswahl abgeschlossen. Ein weiterer wichtiger Aspekt des Testbeds, nämlich das Topologie-Management, wurde ebenfalls durch Beschreibung verschiedener Verfahren abgedeckt. Die Wahl fiel dabei auf ein vielversprechendes Verfahren, welches innerhalb eines EU-Projekts entwickelt wurde. Alle bis dahin ausgewählten Betriebssysteme, Protokolle und Verfahren wurden noch einmal in Abschnitt 2.5 zusammengetragen.

3 Aufbau des statischen Testbeds

In diesem Kapitel werden zunächst alternative Funktechnologien und Sensormethoden geprüft und beschrieben. Anschließend erfolgt ein Überblick über den Laboraufbau bzw. die Realisierung des in Kapitel 2 beschriebenen Konzepts. Dabei werden die aktuellen Anwendungen beschrieben und es wird insbesondere erläutert, an welchen Stellen das Konzept aufgrund praktischer Erfahrungen angepasst werden musste.

3.1 Alternative Funktechnologien

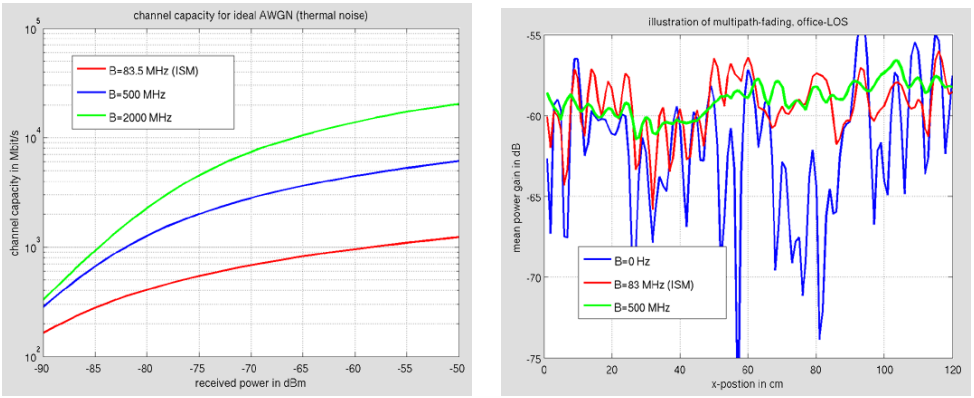
Neben dem IEEE 802.15.4 Standard für drahtlose Kommunikation existieren weitere Möglichkeiten der Kommunikation. Die Vor- und Nachteile verschiedener Möglichkeiten in Bezug auf Kommunikation in Sensornetzen sollen in diesem Abschnitt aufgezeigt werden.

3.1.1 Wireless Local Area Network und Bluetooth

Wie bereits in Abschnitt 2.1.1 beschrieben, eignen sich beide Funktechniken (WLAN und Bluetooth) nur bedingt für Sensornetze. Wenngleich die Datenrate von WLAN mit 54 Mbps vergleichsweise hoch ist, so ist auch der Energieverbrauch eines WLAN-Funkchips deutlich größer. Ähnliches trifft auf Bluetooth zu, dessen Datenrate 3 Megabit per second (Mbps) beträgt. Der hohe Energiebedarf steht im Widerspruch zu energiesparenden Sensorknoten, wodurch sich weder WLAN noch Bluetooth für die Kommunikation zwischen Sensorknoten eignet.

3.1.2 Ultra Wide Band

Eine weitere Alternative bietet die Kommunikation durch Frequenzen im Ultra Wide Band (UWB). Die Bandbreiten bewegen sich hierbei im Bereich von mehreren hundert Megahertz bis hin zu mehreren Gigahertz. Die Sendeleistung ist bei der UWB-Kommunikation äußerst gering, was sich in Bezug auf Sensorknoten als hilfreich erweisen kann. Allerdings ist diese Technik der Kommunikation noch sehr neu. So neu, dass erst Anfang 2008 die Nutzung der UWB-Frequenzen für den privaten Bereich von der Bundesnetzagentur geregelt wurde [29]. Entsprechend existieren bisher keine Lösungen, die kommerziell verfügbar sind.



(a) Datenrate in Abhängigkeit der Empfangsleistung

(b) Empfangsleistung in Abhängigkeit von der Position

Abbildung 3.1: Auswirkungen der Bandbreite auf die Datenrate und die Effekte des Multipath-Fading (Quelle: [168]).

Das DFG-Schwerpunktprogramm Ultra-Wideband Radio Technologies for Communications, Localization and Sensor Applications (UKoLoS) der TU Ilmenau hat zum Ziel, die realen Nutzungsmöglichkeiten von UWB zu erforschen und auf lange Sicht nutzbar zu machen. Das Projekt umfasst neben dem Kommunikationsaspekt ebenfalls die Nutzung von UWB zur Lokalisierung sowie den Einsatz als Sensortechnologie, um bspw. Risse in Gesteinsschichten zu erkennen. Hierfür sei UWB den schmalbandigeren Technologien in mehreren Punkten überlegen [168]. Zunächst lassen sich trotz geringem Energieverbrauch hohe Datenraten erzielen. Da für die Kommunikation in Sensornetzen ohnehin keine überaus großen Datenraten benötigt werden, lässt sich der Energieverbrauch weiter reduzieren. Abbildung 3.1(a) veranschaulicht diesen Sachverhalt [168]. Zu sehen ist die mögliche Übertragungsrate in Abhängigkeit der Empfangsleistung (und somit indirekt auch in Abhängigkeit der Sendeleistung) für verschiedene Bandbreiten. In rot dargestellt ist die Bandbreite, die vom IEEE 802.15.4 Standard verwendet werden kann und die eine geringe Datenrate erreicht. Blau und grün kennzeichnen eine breitbandigere Kommunikation, durch die bei deutlich geringerer Leistung die gleiche Übertragungsrate erzielt werden kann. Zusätzlich werden durch die deutlich höhere Bandbreite die Effekte von Multipath-Fading reduziert. Abbildung 3.1(b) zeigt die Empfangsleistung in Abhängigkeit der Position für verschiedene Bandbreiten. Während sehr schmalbandige Kommunikation, darunter auch Bandbreiten, wie sie in IEEE 802.15.4 verwendet werden, stark bis sehr stark von Multipath-Fading beeinträchtigt wird, zeigen sich diese Effekte für breitbandigere Verbindungen nur noch in abgeschwächter Form. Folglich fallen Schwankungen der Empfangsleistung hier deutlich geringer aus. Ein positiver Nebeneffekt, der gerade in sicheren Sensornetzen von Bedeutung ist, ist die erhöhte Resistenz gegen Jamming (siehe [168]). Als Konsequenz der geringen Sendeleistung und der besseren Jamming-Resistenz braucht bei UWB-Kommunikation kaum Rücksicht auf die derzeitige Verwendung des Frequenzbands genommen werden. Standards wie z. B. WLAN oder Bluetooth arbeiten

mit deutlich höheren Sendeleistungen, so dass die UWB Kommunikation hier nur als Rauschen wahrgenommen und ignoriert wird.

Wie bereits erwähnt eignet sich UWB nicht nur für die Kommunikation, sondern auch zur Lokalisierung. Eine Kombination von Lokalisierung und Kommunikation zur Unterstützung des Routings ist denkbar, jedoch ist auch hier die Forschung noch in ihren Anfängen (siehe [169]).

Insgesamt ist UWB eine interessante und vielversprechende Lösung sowohl für Kommunikation als auch für Lokalisierung. Leider ist sie aber derzeit noch nicht verfügbar.

3.1.3 EnOcean

Bei EnOcean handelt es sich um eine Technik, bei der Sensorknoten bereits durch die Veränderung der Messwerte, wie z.B. eine Temperatur- oder Lichtstärkenänderung, mithilfe entsprechender Wandler Energie aufnehmen. Zusätzlich wird verfügbare Energie der Umgebung (z.B. Sonnenlicht) verwendet. Demzufolge benötigen die Sensoren keine Batterie. Sie basieren auf sogenanntem Energy-Harvesting. Die so gewonnene Energie wird auch verwendet, um die gemessenen Werte drahtlos zu einer Senke zu übermitteln. Die Datenübertragung wird ebenfalls mit Hinblick auf einen möglichst geringen Stromverbrauch realisiert. Hierfür wird einerseits nur für Bits mit Wert 1 ein Signal übertragen. Andererseits werden Pakete nicht sofort, sondern mit einer pseudozufälligen Zeitverschiebung versendet, um Kollisionen möglichst zu vermeiden. Die erzielte Datenrate liegt bei 120 kilobits per second (kbps) [51].

EnOcean legt seinen derzeitigen Schwerpunkt auf die drahtlose Bedienung von Komponenten, wie z. B. das Steuern von Lichtsystemen. In Zukunft wird EnOcean jedoch zunehmend interessanter für WSNs, wenn auch eine Interoperabilität mit derzeitigen WSNs nicht einfach zu bewerkstelligen ist.

3.2 Alternative Sensormethoden

In diesem Abschnitt werden Sensortechnologien vorgestellt, mit denen sich Informationen der Umgebung sammeln lassen.

3.2.1 Ultraschall

Ultraschall kann zur Bestimmung von Position und Größe von Objekten verwendet werden. Das Funktionsprinzip ist dabei identisch zu dem von einem Sonar: es werden akustische Signale ausgesendet, die durch Objekte reflektiert und wieder empfangen werden. Die empfangenen Signale geben Aufschluss über die Größe, Form und Entfernung des Objekts.

Derzeit existieren entsprechende Sensoren vor allem im Bereich der Robotik oder auch in der Medizin. Für Sensorknoten ist Ultraschall als Sensortechnologie derzeit ein aktives Forschungsgebiet. Das in Abschnitt 3.1.2 beschriebene Projekt UKoLoS [168] versucht UWB nicht nur zu Kommunikationszwecken, sondern auch zur Lokalisierung einzusetzen. Die in UKoLoS durch UWB verwendete hohe Bandbreite führt dazu, dass die Lokalisierung sehr genau ist. Die im Spektrum von UWB enthaltenen niedrigen Frequenzen führen außerdem dazu, dass eine Lokalisierung auch möglich ist, wenn kein Sichtkontakt besteht. Folglich könnten Sensoren hinter einem Hindernis bereits Personen erkennen, bevor diese das Hindernis überwunden haben.

Ultraschall hält ebenso Einzug in den Bereich von WSNs. So gibt es von der Firma MEMSIC ein entsprechendes Modell, das auf der MICA2 Hardware aufbaut und zusätzlich mit einem Ultraschallsender und -empfänger bestückt ist [105]. Zusammen mit einem Radio Frequency (RF)-Transceiver erlaubt dies dem Sensorknoten die Entfernung zu anderen derartigen Sensorknoten zu bestimmen. Hierfür senden einige Knoten gleichzeitig einen RF- und einen Ultraschallimpuls aus. Empfängt ein anderer Knoten den RF-Impuls, kann aus dem zeitverschobenen Eintreffen des Ultraschallimpulses die Entfernung zu dem dazugehörigen Sender bestimmt werden. An dieser Stelle sei angemerkt, dass neuere iSense-Knoten (mit Funkmodul Jennic JN5148) eine Entfernungsbestimmung auf Basis von Signalumlaufzeiten ermöglichen.

3.2.2 Laserbasierte Ansätze

Ähnlich zur im vorherigen Abschnitt beschriebenen Anwendung von Ultraschall zur Lokalisierung kann auch ein Laser zum gleichen Zweck eingesetzt werden. Die Funktionsweise unterscheidet sich hierbei dadurch, dass Laser sehr punktuelle Messungen erlauben. Durch feine Änderungen der Ausrichtung des Lasers lässt sich ein Bereich abtasten, wodurch eine Abbildung der Umgebung entsteht. Dieser Prozess wird in der Regel sehr schnell durchgeführt, so dass eine komplette Abbildung in Bruchteilen von einer Sekunde erstellt werden kann [76]. Je nach Umsetzung kann dieser Prozess jedoch derzeit sehr energieintensiv sein. Neben dem häufigen Abtasten spielt vor allem die Berechnung eine Rolle, mit der die verschiedenen Messungen einer Abbildung aggregiert werden. Typische Sensoren, wie sie in der Robotik eingesetzt werden, haben daher einen Verbrauch von 2 bis 15 Watt [77].

3.2.3 Fazit

Die hier vorgestellten Sensormethoden bieten im Vergleich zu gängigen Sensorknoten einige Vorteile und Möglichkeiten. Die Entwicklung ist jedoch meist noch am Anfang – es sind keine entsprechenden Komponenten erhältlich. Eine Lokalisierung von unbekannten Objekten zu ermöglichen ist derzeit nur möglich, indem Sensortechnologie, die für die Robotik oder andere Bereiche entwickelt wurde, entsprechend für Sensorknoten umgebaut wird. Dabei ist allerdings die

Ressourcen-Beschränkung von Sensorknoten, wie z.B. der Energieverbrauch, zu berücksichtigen.

3.3 Beschreibung des Laboraufbaus

Nach dem Überblick über alternative Funktechnologien und Sensormethoden wird nun der Laboraufbau beschrieben. Dabei wurde nach dem Konzept (vgl. Abschnitt 2.5.2) ein statisches Sensornetz aufgebaut, das in drei Teilnetze unterteilt ist, denen je eines der drei WSN-Betriebssysteme (Contiki [47], iSense [33] und TinyOS [74]) zugeordnet ist. Das Verwenden unterschiedlicher Funkkanäle innerhalb dieser Teilnetze stellt dabei den interferenzfreien parallelen Betrieb der Netze sicher. Des Weiteren wurde ein Laptop als zentrale Basisstation konfiguriert und über die Universal Asynchronous Receiver Transmitter (UART)-Schnittstelle mit mehreren Sensorknoten verbunden, die als Gateway zwischen der Basisstation und den Sensornetzen fungieren.

Das iSense-Netz wurde als homogenes Netzwerk umgesetzt, das sich aus den iSense Core Modulen CM10C mit Security Modulen zusammensetzt, die über einen Beschleunigungssensor und einen passiven Infrarot-Sensor (PIR) verfügen. Als Routing-Protokoll im iSense-Teil wird aktuell ein baumbasiertes Protokoll verwendet.

Das TinyOS-Netz wurde dem Konzept entsprechend als heterogenes Netzwerk realisiert, das sich aus den beiden Plattformen TelosB und MicaZ zusammensetzt. Während auf der TelosB-Plattform Sensoren zur Messung von Temperatur, Lichtstärke und Luftfeuchtigkeit integriert sind, verfügt die MicaZ-Plattform über keinerlei Sensorik und muss analog zu dem iSense Core Module durch das Anstecken eines Sensorboard-Moduls um diese erweitert werden. Als geeignetes Sensorboard wurde das MTS400 ausgewählt, mit dem die Sensortypen des TelosB-Knotens abgedeckt werden. Das TinyOS-Netz wird mit dem CTP-Routingprotokoll betrieben.

Entgegen dem Konzept war es nicht möglich, das Contiki-Netz ebenfalls als heterogenes Netzwerk bestehend aus TelosB- und MicaZ-Knoten zu realisieren. Im Rahmen der praktischen Arbeiten zeigte sich, dass das Betriebssystem Contiki in der aktuellen Version keine vollständige Unterstützung der MicaZ-Plattform bietet. Somit ist der Einsatz dieser Plattform nicht uneingeschränkt gewährleistet. Unter Contiki ist es derzeit nicht möglich, auf den Flash-Speicher eines MicaZ-Knotens zuzugreifen. Diese Operation wird jedoch zwingend für das lokale, nichtflüchtige Speichern von Informationen und insbesondere für die OTAP-Anwendung benötigt. Aus diesem Grund wurde zunächst auf die Heterogenität des Contiki-Netzes verzichtet und sich auf den Einsatz von TelosB-Knoten in diesem Netzwerk beschränkt. Im Contiki-Netz wird als Routingprotokoll derzeit RPL verwendet, wobei eine Implementierung des CTP-Protokolls ebenfalls vorhanden ist.

Teilnetz	OS	Hardware	Kanal
1	Contiki	10× TelosB	11
2	iSense	10× Core Module CM10C mit Security Module	18
3	TinyOS	5× TelosB und 5× MicaZ mit MTS400-Board	26

Tabelle 3.1: Hardware-Spezifikation und Funkkanal der drei realisierten Teilnetze.

Die aktuellen in den drei Teilnetzen verwendeten Hardware-Komponenten und deren Stückzahlen sind zusammenfassend in Tabelle 3.1 aufgelistet. Darüber hinaus ist ein Überblick, der die Spezifikationen aller verwendeten Hardware-Komponenten enthält, in Abschnitt 2.1.2 zu finden.

Im Folgenden wird zunächst die Sensornetz-Applikation vorgestellt und anschließend auf das OTAP sowie das Topologie-Management in der Testumgebung eingegangen. Danach werden der Paket-Sniffer, der zur Analyse und zur Visualisierung des Testbeds dient, und der IEEE 802.15.4 Jammer, mithilfe dessen sich Jamming-Angriffe realisieren lassen, vorgestellt.

3.3.1 Sensornetz-Applikation

Als repräsentative Sensornetz-Applikation wurde sich für eine Anwendung entschieden, die der Klasse des kontinuierlichen Data Delivery Modells zuzuordnen ist und in der Convergecasts das primäre Kommunikationsmuster bilden (vgl. Abschnitt 2.2.1). Das bedeutet konkret, dass für jedes Betriebssystem bzw. Teilnetz eine Anwendung basierend auf den dazu korrespondierenden Routing-Protokollen implementiert wurde. In diesen Anwendungen wird die Sensorik der entsprechenden Hardware-Plattformen (siehe Tabelle 3.1) periodisch (alle 5 s) ausgelesen. Die gesammelten Informationen werden anschließend an den Gateway-Knoten übertragen, der dem entsprechenden Teilnetz zugeordnet und mit der Basisstation verbunden ist. Dieser bildet anhand einer speziellen Applikation die Datensenke des Teilnetzes und ermöglicht die Weiterleitung der Daten an die zentrale Basisstation. Für die weitere Verarbeitung der Sensordaten ist die Basisstation zuständig. Einerseits ist es möglich, diese Daten online zu visualisieren, andererseits können diese für eine spätere Offline-Analyse abgespeichert werden.

3.3.2 Over-The-Air Programming

Entsprechend dem Konzept über die einzusetzenden OTAP-Protokolle in Abschnitt 2.5.1 wurde im iSense-Netz auf die vorhandene Anwendung zurückgegriffen. Sowohl erste Funktionstests als auch der erfolgreiche Einsatz im realisierten Sensornetz bestätigten die fehlerfreie Funktion dieser Anwendung.

Im Contiki- und TinyOS-Netz wurde in Abschnitt 2.3.2 der Einsatz des OTAP-Protokolls Deluge vorgesehen. Analog zu der OTAP-Anwendung von iSense wurden beide Implementierungen im praktischen Einsatz getestet. Beim Testen

der TinyOS-Version ergaben sich unerwartete Schwierigkeiten. Es hat sich gezeigt, dass eine zuverlässige Ausführung einer OTAP-Operation in Netzwerken der vorgesehenen Größe mit der aktuellen Version nicht gewährleistet ist und oftmals nur eine zufällige Teilmenge der Knoten mit dem propagierten Binärcode reprogrammiert wird. In einer Vielzahl von Fällen fordern Knoten, deren Reprogrammierung erfolglos war, nach einem Neustart den Binärcode vergeblich erneut an und wiederholen diesen Versuch, sobald die wiederholte Übertragung des Codes beendet ist. Bei dem Versuch, die Contiki-Implementierung von Deluge einzusetzen, ergab sich, dass diese aktuell nur eingeschränkt lauffähig ist. Zwar wird die Verbreitung des auszuführenden Binärcodes in dieser Implementierung unterstützt, die Komponente, die zur eigentlichen Ausführung dieses Updates notwendig ist, ist jedoch derzeit nicht in die Deluge-Implementierung integriert.

Zudem ergab sich bei intensiverer Beschäftigung mit Deluge, dass für eine Testbed-Umgebung eine derartig komplexe und fehleranfällige Anwendung nicht sinnvoll ist. Aus diesen Gründen wird eine einfache OTAP-Basisanwendung realisiert. In dieser Anwendung wird auf den komplexen und aufwendigen Verteilungs-Algorithmus, den Deluge verwendet, verzichtet. Die neue Anwendung wird sowohl für Contiki als auch für TinyOS das gezielte Reprogrammieren ausgewählter Sensorknoten ermöglichen.

Neben den praktischen Erfahrungen zur Einsetzbarkeit von Deluge wurde anhand der Funktionstests der vorhandenen Implementierungen deutlich, dass bereits die OTAP-Anwendung an die Ressourcengrenze der Sensor-Hardware stößt. Programm- und Arbeitsspeicherressourcen sowohl der TelosB- als auch der MicaZ-Plattform sind nicht ausreichend, um bspw. die Datensenke der realisierten Anwendung sowie die OTAP-Basisstation zugleich auf den Knoten zu realisieren. Ebenso die Kombination des Anwendungsteils für die einzelnen Sensorknoten mit der Deluge-Komponente für das Ausführen des Binärcodes lastet die verfügbaren Ressourcen nahezu aus.

3.3.3 Topologie-Management im Testbed

In dem Konzept für das Topologie-Management des Testbeds (siehe Abschnitt 2.4) wurde der Ansatz der *Virtual Links* ausgewählt, der im Rahmen des EU-Projekts WISEBED [40, 148] entwickelt wird und im Gegensatz zu alternativen Methoden eine bessere Reproduzierbarkeit und ein komfortableres Topologie-Management verspricht. Anstelle einer Adaptionsschicht eines WSN-Betriebssystems für einen Simulator und weitere Hardwareplattformen zu entwickeln, generalisiert die Wiselib (vgl. [16]) dieses Konzept hin zu beliebigen Hard- und Softwarezielplattformen. Die Wiselib ist eine Algorithmenbibliothek und ein Abstraktionslayer zur Vereinfachung der Applikationsentwicklung. Eine Implementierung des Konzepts der Virtual Links ist in der Wiselib verfügbar.

Die Wiselib basiert auf C++ Techniken, wie z.B. Templates und Inline Functions, die es erlauben, generischen Programmcode zu schreiben. Des Weiteren bie-

tet dieser Ansatz die Möglichkeit, dass der Compiler Optimierungen vornehmen kann, die bei klassischer Programmierung mit objektorientierten Ansätzen nicht möglich sind. Das Resultat ist optimierter, kompakter und schneller Binärcode, der auf Sensorknoten und auch in Simulationsumgebungen ausgeführt werden kann.

Der aktuelle Stand der Wiselib ist, dass die Simulationsumgebung Shawn und die Hardwareplattform iSense gut unterstützt werden. An der vollständigen Unterstützung anderer Zielplattformen, wie u. a. Contiki und TinyOS, wird derzeit gearbeitet.

Für das Topologie-Management basierend auf virtuellen Links konnte somit bei allen Plattformen (wie geplant) auf die Wiselib-Implementierung zurückgegriffen werden. Daher werden entsprechende native Virtual Radio Software-Komponenten für die Betriebssysteme Contiki und TinyOS implementiert. Bei der Implementierung werden ausschließlich für dieses Projekt relevante Kernfunktionalitäten der Wiselib-Implementierung, die (De-)Aktivierung selektierter, physikalischer Netzwerkverbindungen, realisiert.

Eine weitere notwendige Anpassung der Wiselib-Implementierung betrifft den Testbed Server. Dieser setzt die Verkabelung sämtlicher Sensorknoten voraus, um diese für das Topologie-Management via UART-Schnittstelle nutzen zu können. Weil in der in diesem Projekt realisierten Testumgebung jedoch nicht die Verkabelung aller Hardware-Komponenten vorgesehen ist, muss der Server für das drahtlose Topologie-Management erweitert werden.

3.3.4 Paket-Sniffer

Neben den Gateway-Knoten, die über die UART-Schnittstelle mit dem als Basisstation konfigurierten Laptop kommunizieren, wurden drei Paket-Sniffer mit diesem Laptop verbunden. Wie in Abschnitt 2.5.2 beschrieben, handelt es sich bei einem derartigen Paket-Sniffer um einen mit Contiki-Code modifizierten AVR RZUSBSTICK von Atmel [12]. Sämtlicher Datenverkehr in den oben beschriebenen Teilnetzen kann mittels der Paket-Sniffer überwacht und aufgezeichnet werden. Zur Visualisierung dieser Daten wird das Protokollanalyse-Tool Wireshark [178] verwendet. Ein Screenshot der Wireshark-Benutzeroberfläche, der den mithilfe des Sniffers aufgezeichneten Datenverkehr im TinyOS-Netzwerk darstellt, ist in Abbildung 3.2 zu sehen.

In Wireshark werden die vom Sniffer empfangenen Pakete unmittelbar angezeigt und dabei zeitlich aufbauend als Paketstrom dargestellt (oberes Fenster des Screenshots). Neben der binären und hexadezimalen Darstellung der Pakete (unteres Fenster des Screenshots), ist es möglich Wireshark um sogenannte Dissectoren zu erweitern. Diese erlauben dem Tool, die einzelnen Bytes der in dem Bytestrom eines Pakets enthaltenen Protokollfelder verschiedener Ebenen korrekt zu interpretieren. Ein IEEE 802.15.4-konformer Dissector ist bereits in Wireshark vorhanden und erleichtert die Analyse der Datenströme auf physikalischer Ebene. Zudem wurden im Rahmen des Projekts diverse WSN-spezifische

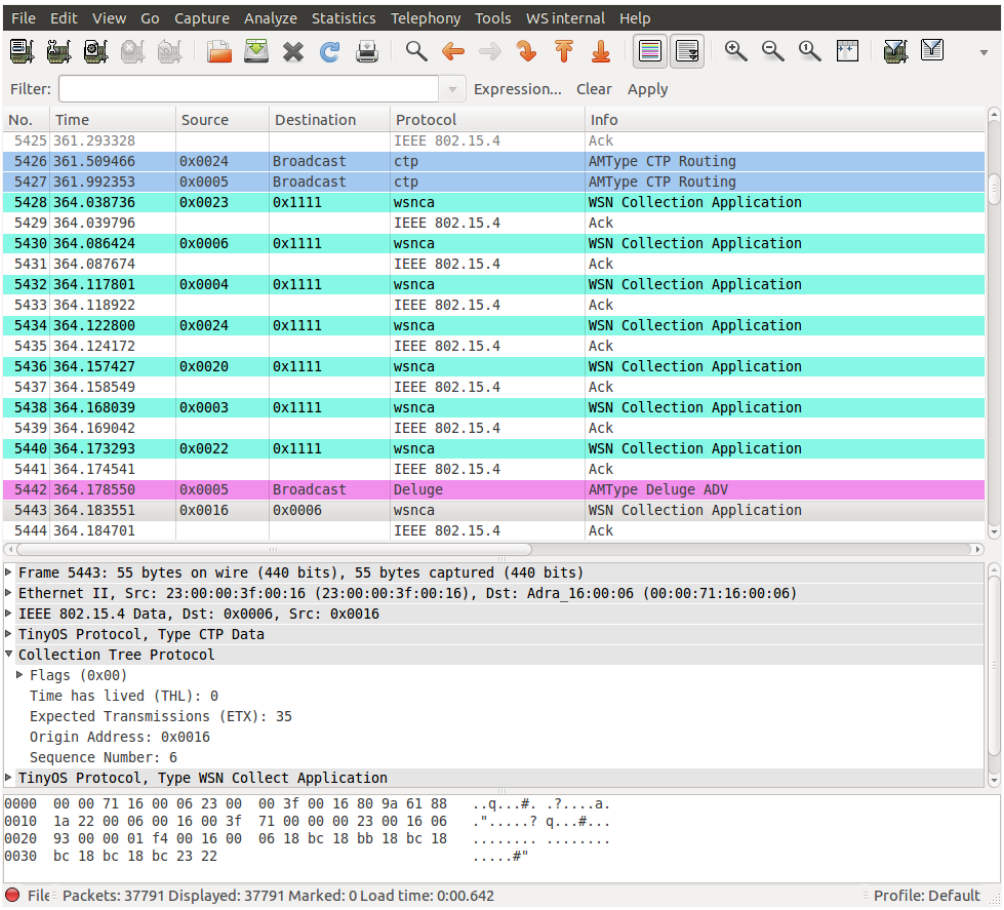


Abbildung 3.2: Wireshark-Screenshot eines interpretierten Paketstroms in dem auf TinyOS basierenden Sensornetz.

Dissectoren implementiert. Beispielsweise zeigt der Screenshot in Abbildung 3.2 die erfolgreiche Interpretation verschiedener TinyOS-Pakete (oberes Fenster des Screenshots) und die aufgelösten Belegungen aller in einem selektierten Paket enthaltener Protokollfelder (mittleres Fenster des Screenshots).

3.3.5 IEEE 802.15.4 Jammer

Wie im Konzept (vgl. Abschnitt 2.5.2) vorgesehen, wurde auf einer SDR-basierenden Komponente ein Jammer für den Angriff auf IEEE 802.15.4-konforme Drahtlosnetzwerke realisiert, der sich per Remote-Verbindung kontrollieren lässt. Dabei wurde auf das Universal Software Radio Peripheral (USRP)-Board [55] mit dem GNU Radio SDK [63] zurückgegriffen.

Erste Tests des Jammers haben gezeigt, dass durch das Jamming in der Testumgebung die Kommunikation zwischen den Sensorknoten und der Basisstation erfolgreich unterbunden werden kann. Der Screenshot in Abbildung 3.3 demonstriert diese Situation. Die im TinyOS-Sourcecode verfügbare Oscilloscope-Anwendung wurde hier auf der Basisstation ausgeführt. Sie visualisiert exem-

Abbildung 3.3: Screenshot der TinyOS-Oscilloscope-Applikation. Während der Jamming-Phase (rot umrandeter Bereich) wird der Sensordatenstrom zur Basisstation unterbunden.

plarische Sensordaten von fünf Knoten, die anhand der Färbung differenziert werden können, indem die Datenwerte (y-Achse) über die Zeit (x-Achse) aufgetragen werden. Die Daten wurden dabei in der Sensornetz-Applikation des TinyOS-Netzes generiert und an die Basisstation weitergeleitet. Der rot umrahmte Bereich hebt die Jamming-Phase hervor, die den Empfang der Sensordaten an der Basisstation sichtlich verhindert. Nach Deaktivierung des Jammers kann die Kommunikation fortgesetzt werden.

3.4 Zusammenfassung

In diesem Kapitel wurden zunächst alternative Funktechnologien (Abschnitt 3.1) und Sensormethoden (Abschnitt 3.2) geprüft und beschrieben. Bei den Funktechnologien stellt UWB eine sehr interessante und vielversprechende Lösung dar, die neben der Kommunikation auch zur Lokalisierung verwendet werden kann. Der aktuelle Stand der Technik ist allerdings für einen direkten Einsatz im Rahmen dieses Projekts noch nicht ausreichend. EnOcean ist gerade aufgrund des realisierten Energy-Havestings eine vielversprechende Alternative. Eine Integration in bzw. Interoperabilität mit derzeitigen Sensorknoten ist allerdings bisher nicht verfügbar. Auch die betrachteten Sensormethoden (Ultraschall und laserbasierte Ansätze) sind interessante Möglichkeiten zur Lokalisierung, wobei der Einsatz im Testbed nicht einfach möglich ist.

Im Anschluss an die Sichtung der alternativen Technologien erfolgte ein Überblick über den Laboraufbau bzw. die Realisierung des in Kapitel 2 beschriebenen Konzepts. Dabei wurde insbesondere erläutert, an welchen Stellen das Konzept aufgrund praktischer Erfahrungen angepasst werden musste. Verschiedene Protokolle und Tools, die laut Entwicklern stabil und fehlerfrei funktionieren sollten, waren (noch) nicht ausgereift genug. Darüber hinaus zeigt sich, dass die Ressourcen-Beschränktheit der Knoten einen erheblichen Einfluss auf die Realisierung hat. Es werden daher Alternativen für das OTAP und das Topologie-Management realisiert.

Abbildung 3.4: Foto des Testbeds.

Ein Foto des umgesetzten Testbeds ist in Abbildung 3.4 zu sehen. Die drei Tische im Hintergrund sind den Teilnetzen Contiki, TinyOS und iSense (von links nach rechts) zugeordnet, während im Vordergrund der Server mit Jammer und Sniffern aufgebaut ist.

4 Erweiterung des Testbeds um Mobilität

Neben statischen Sensornetzen kommt auch mobilen Sensornetzen eine immer größere Bedeutung zu. Die experimentelle Evaluation innerhalb eines Testbeds ist jedoch ungleich schwerer durchzuführen, insbesondere bei komplexeren Bewegungsabläufen der Sensorknoten. In diesem Kapitel wird die Mobilitätserweiterung des Testbeds vorgestellt. Dazu wird zunächst in Abschnitt 4.1 die Gesamtarchitektur im Überblick beschrieben. Darauf folgen weitere Details zur Umsetzung des Virtual Links Konzepts (Abschnitt 4.2) und der dynamischen Topologien (Abschnitt 4.3). Wesentliches zur praktischen Realisierung des Gesamtkonzepts wird dann in Abschnitt 4.4 ausgeführt. Abschließend wird die Mobilitätserweiterung in Abschnitt 4.5 zusammengefasst.

4.1 Gesamtarchitektur

Wie bereits in Abschnitt 2.4 und 3.3.3 beschrieben, kann das Konzept der Virtual Links [15] dazu verwendet werden, beliebige Topologien in einem Testbed zu definieren und umzusetzen. Die gewünschte Topologie wird zu Beginn der Evaluation durch das gesamte Netz propagiert, so dass jeder Sensorknoten die Routingtabelle seines Virtual Radios entsprechend anpassen kann. Das WiseML-Format [149], welches zur Definition der virtuellen Topologie dient, unterstützt außerdem Zeitstempel, die es ermöglichen, Links zu einem bestimmten Zeitpunkt während des Experiments ein- oder auszuschalten. Damit ist es also möglich, die Topologie im laufenden Betrieb zu ändern.

Der Zusammenhang zum mobilen Sensornetz besteht nun darin, dass sich bei (ausreichend hoher) Mobilität neben den Knotenpositionen auch die Topologie ändert. Entfernt sich ein Knoten bspw. weit genug von seinem direkten Nachbarn, so reißt der Link zwischen diesen ab. Um bestimmen zu können, zu welchem Zeitpunkt sich die Mobilität auf die Topologie auswirkt, wird ein Signalausbreitungsmodell benötigt: Mit diesem kann berechnet werden, wann sich ein Knoten außerhalb des Kommunikationsradius eines anderen Knotens befindet, d.h. wann der Link zwischen den Knoten abbricht. Verknüpft man das Signalausbreitungsmodell mit einem Bewegungsmodell, lässt sich demnach über die Zeit hinweg berechnen, wie sich die Topologie auf Basis der Knotenmobilität ändert.

Die grundlegende Konzeptidee ist also, mithilfe eines Bewegungsgenerators und Signalausbreitungsmodells eine WiseML-konforme Datei zu erzeugen, in der die

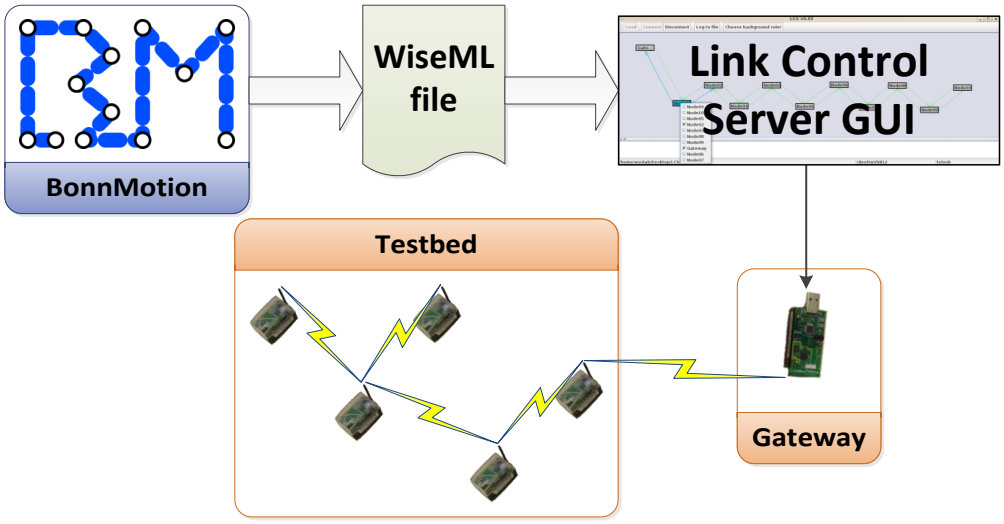


Abbildung 4.1: Schematische Darstellung des Konzepts des mobilen WSNs.

Topologieänderungen definiert sind und die als Input für die Umsetzung der Virtual Links verwendet werden kann. Abbildung 4.1 stellt das Konzept noch einmal schematisch dar: Zunächst wird mit dem Bewegungsgenerator BonnMotion eine WiseML-Datei erzeugt, welche Informationen über die dynamische Topologie enthält. Diese Datei dient als Input für die graphische Oberfläche des Link Control Servers (LCSs) (Details unten). Der Server verarbeitet die Daten weiter und schickt sie in geeignetem Format via UART an das Link Control Gateway. Im letzten Schritt werden die Kommandos zum Ein- bzw. Ausschalten eines Links an die betroffenen Knoten im Testbed drahtlos übertragen.

4.2 Link Control

Die nativen Virtual Radio Komponenten wurden, wie in Abschnitt 3.3.3 beschrieben, implementiert. Dazu war es nötig, für jedes Betriebssystem eine Client-Komponente (Virtual Radio) sowie eine Gateway-Komponente zu implementieren. Da es bei der Implementierung nicht notwendig war, virtuelle Links zu erschaffen, sondern lediglich reale Links ein- bzw. auszuschalten, wurde die treffendere Bezeichnung „Link Control“ für diese Umsetzung des Konzepts der Virtual Links gewählt. Die zentrale Server-Komponente ist unabhängig vom Sensor-Betriebssystem und mithilfe einer graphischen Oberfläche steuerbar (siehe Abbildung 4.2). Das Zusammenspiel zwischen den einzelnen Komponenten läuft dabei wie folgt ab:

Zunächst wird vom Server eine serielle Verbindung (UART) zu einem der Gateways hergestellt. Darauf werden die Topologiedaten aus einer Datei eingelesen und in Form von `enablePhysicalLink` / `disablePhysicalLink` Kommandos an das Gateway via UART übertragen. Das Gateway sendet diese Kommandos drahtlos via single-hop an die betroffenen Knoten weiter. Sobald ein Knoten ein Kommando empfangen hat, ändert dieser Empfangs- bzw. Sende-

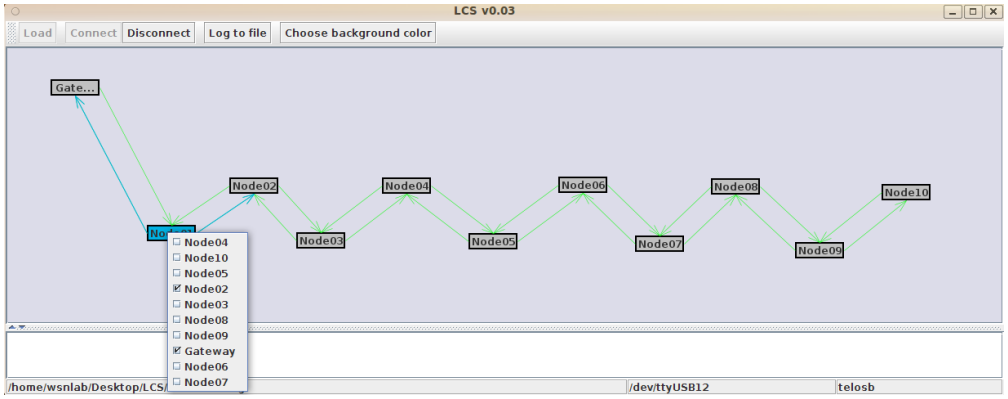


Abbildung 4.2: Link Control Server GUI.

Whitelist seines Virtual Radios entsprechend und quittiert den Empfang mit einem Acknowledgement. Ein ähnlicher Best Effort Service war im Rahmen der Wiselib-Implementierung nicht nötig, da sich diese in ihrer aktuellen Version auf Testbeds beschränkt, bei denen alle Knoten mit dem Server verkabelt sind.

Wie oben bereits erwähnt, pflegt jeder Knoten für jede Kommunikationsrichtung eine Whitelist. Diese beiden Whitelists sind zu Beginn leer, so dass nur Broadcast-Nachrichten versendet und Link-Control-Nachrichten vom Gateway empfangen werden können. Die Kommandos einer Link-Control-Nachricht beziehen sich jeweils nur auf eine Senderichtung eines Knotenpaares, womit unidirektionale Links ermöglicht werden. Sobald ein Paket an einen Empfänger versendet werden soll, der nicht in der Sende-Whitelist enthalten ist, wird das höchste Bit der Empfängeradresse gesetzt. Obwohl das modifizierte Paket auf diese Weise nicht beim ursprünglich vorgesehenen Empfänger ankommt, wird es trotzdem übertragen. Die Wiselib-Implementierung sieht dies nicht vor und verwirft ein solches Paket. Im Kontext der virtuellen Mobilität erschien es jedoch sinnvoll, die den dadurch entstehenden Effekt der Interferenz beizubehalten, denn ein mobiler Knoten unterbricht in der Realität auch nicht abrupt seine Sendeversuche, sobald er außerhalb der Übertragungsreichweite des Empfängers ist. Wird andersherum ein Paket von einem Sender empfangen, der nicht auf der Empfangs-Whitelist steht, wird dieses verworfen. Eine Ausnahme bilden hier lediglich die Link-Control-Nachrichten, welche immer verarbeitet werden.

4.3 Dynamische Topologien

Ein Werkzeug zur Umsetzung beliebiger (virtueller) Topologien ist mit der Link-Control-Implementierung vorhanden. Für die Gesamtarchitektur des mobilen Sensornetzes fehlt demnach noch ein Werkzeug zum automatischen Generieren dynamischer Topologien. Für diese Aufgabe eignet es sich, den Bewegungsszenariogenerator BonnMotion [8] entsprechend zu erweitern.

BonnMotion ist ein Java-Programm, mit dem Bewegungsszenarien erzeugt und analysiert werden können. Es werden diverse zufalls-basierte sowie szenario-

spezifische Bewegungsmodelle unterstützt. Weiterhin können die generierten Bewegungsszenarien in verschiedene Formate exportiert werden, so dass sie von entsprechenden Netzwerk-Simulatoren oder anderen Programmen weiterverarbeitet werden können. Die Bewegung der Knoten wird nativ in Form von sogenannten *Waypoints* abgespeichert. Ein Waypoint kann als Tripel $(id, time, pos)$ angesehen werden, wobei

- id die Knoten-ID,
- $time$ die Simulationszeit (Zeit relativ zu Beginn des Szenarios) und
- pos die Position innerhalb der spezifizierten Simulationsfläche

bezeichnet. Der vollständige Bewegungsablauf eines Knotens kann nun bestimmt werden, indem jeweils zwei chronologisch aufeinanderfolgende Waypoints desselben Knotens durch eine Gerade verbunden werden. Auf dieser Gerade ist die Geschwindigkeit des Knotens konstant und kann aus Distanz und Zeitdifferenz berechnet werden.

Die oben beschriebene Waypoint-Darstellung liefert für sich alleine allerdings noch keine Informationen über Linkentstehung und -abbruch, welche für die Spezifikation dynamischer Topologien benötigt werden. Daher soll ein WiseML-Exporter implementiert werden, der auf Basis der Waypoints in Kombination mit einem Signalausbreitungsmodell sogenannte *Contacts* berechnet. Ein Contact zwischen zwei Knoten besteht genau dann, wenn ein Link besteht, d.h. wenn sich die Knoten innerhalb des gegenseitigen Kommunikationsradius befinden. Damit kann bestimmt werden, zu welchem Zeitpunkt ein Link besteht bzw. abbricht.

4.4 Realisierung

Um dynamische Topologien mit BonnMotion im WiseML-Format spezifizieren zu können, wurde der bereits vorhandene WiseML-Exporter (vgl. [170, Abschnitt 7.6]) der BonnMotion Version 1.5a entsprechend erweitert. In seiner bisherigen Funktionsweise wurden lediglich in parameter-gesteuerten Zeitabständen alle Knotenpositionen ausgegeben. Dieser intervall-basierte Modus ist jedoch für das Konzept der Link Control ungeeignet. Daher wurde ein weiterer kontakt-basierter Modus hinzugefügt, bei welchem die Contacts berechnet werden. Bei diesem muss der Kommunikationsradius als Parameter (`-r`) übergeben werden, welcher dann als Basis für ein vereinfachtes kreis-basiertes Signalausbreitungsmodell dient. Standardmäßig werden bei diesem Modus alle Knotenpositionen ausgegeben, sobald sich der Status eines beliebigen Links ändert. Mit dem Parameter-Flag `-e` lassen sich zusätzlich zu den Positionen die benötigten `enablePhysicalLink` / `disablePhysicalLink` Kommandos für alle Link-Statusänderungen ausgeben. Um den Overhead zu minimieren, werden mit dem Parameter-Flag `-o` nur die Positionen der Knoten ausgegeben, deren Link-Status sich geändert hat. Zum Zeitpunkt 0 werden alle Links aktiviert, die sich initial anhand der Knotenpositionen und dem Kommunikationsradi-

us ergeben. Dementsprechend werden analog zum Ende des Experiments alle Links deaktiviert, falls sie bis dahin noch aktiviert waren. Mit dieser Erweiterung des WiseML-Exporters lassen sich nun alle von BonnMotion unterstützten Bewegungsmodelle als Basis für das virtuelle mobile Netz verwenden.

Wie bereits weiter oben beschrieben, stellt der LCS eine Verbindung über die serielle Schnittstelle zum Link Control Gateway her. In einer Konfigurationsdatei werden u.a. der serielle Port des Gateways sowie Positionen und Konnektivität der Knoten festgelegt. Handelt es sich um eine statische Topologie, reicht es aus, die Link-Control-Nachrichten einmalig zu Beginn an die betroffenen Knoten zu senden. Um mit dem LCS nun auch dynamische Topologien realisieren zu können, wird in der Konfigurationsdatei der Pfad zur entsprechenden WiseML-Datei spezifiziert. Diese wird zunächst vollständig eingelesen und die initialen Knotenpositionen werden angezeigt. Mit dem Klick auf einen Button startet man die virtuelle Bewegung und die dynamische Topologie wird in Echtzeit abgespielt. Ändert sich der Status eines Links, wird zum dazugehörigen Zeitpunkt das entsprechende Kommando an die betroffenen Knoten versendet. Da, wie oben erklärt, zusätzlich die Knotenpositionen zu diesen Zeitpunkten angegeben werden, können diese auch angezeigt werden. Für flüssige Übergänge bei der Bewegung sorgt eine Interpolation, so dass die Knotenpositionen mit ca. 25 Hz aktualisiert werden. Eine große Herausforderung beim Abspielen ist das Timing: Aufgrund der Verzögerung, die sich aus Nachrichtenübertragung und Bearbeitung des Kommandos zusammensetzt, entsteht ein Offset zwischen WiseML-Zeitstempel und tatsächlichem Zeitpunkt, zu dem das Kommando umgesetzt wird. Da dieser Offset nicht konstant ist und von diversen Einflüssen abhängt, lässt sich die Bewegung also nur näherungsweise zeitgetreu umsetzen.

4.5 Zusammenfassung

In diesem Kapitel wurde die Mobilitätserweiterung des Testbeds vorgestellt. Anstatt auf eine aufwendige physikalische Umsetzung (bspw. durch Roboter) zu setzen, wurde eine software-basierte Lösung zu verfolgt. Das Konzept der Virtual Links lässt sich nicht nur statisch zu Beginn, sondern ebenso dynamisch während eines Experiments anwenden. Da die Linkdynamik der wesentliche Effekt der Knotenmobilität ist, wird letztere simuliert, indem die durch Linkabbruch und -entstehung definierte dynamische Topologie als Input für die Link-Control-Implementierung dient. Um weiterhin beliebige Bewegungsmuster mit diesem Ansatz simulieren zu können, wurde der Bewegungsszenariogenerator BonnMotion um einen entsprechenden Exporter erweitert. So kann auf elegante Weise auf ein breit gefächertes Repertoire an Bewegungsmodellen für das mobile Sensornetz zurückgegriffen werden. Einen zusätzlichen signifikanten Vorteil stellt die Reproduzierbarkeit der Experimente dar, welche aufgrund der lediglich simulierten Mobilität gewonnen wird. Damit können Fehlerquellen schneller identifiziert und Evaluationen statistisch zuverlässig durchgeführt werden.

Teil II

Sicherheitsarchitektur

5 Konzept der Sicherheitsarchitektur

Aufgrund der Vielzahl an Anwendungen von WSNs und insbesondere wegen deren Einsatz im industriellen, medizinischen und militärischen Bereich ist Sicherheit auch in dieser Art von Netzen von hoher Relevanz. In diesem Kapitel wird daher das Konzept einer Sicherheitsarchitektur für WSNs vorgestellt, welches im Projekt umgesetzt wurde. Zunächst wird in Abschnitt 5.1 der Bezug zum BSI-Projekt Modellierung drahtloser Sensornetze (MoSe) hergestellt. Im Anschluss wird eine Gefahrenanalyse (Abschnitt 5.2) durchgeführt. Die darin identifizierten Gefahren werden darauf in den Abschnitten 5.3, 5.4 und 5.5 strukturiert anhand der OSI-Ebenen PHY, MAC bzw. Routing, samt geplanter Gegenmaßnahmen beschrieben. Abschnitt 5.6 stellt weiterhin Angriffe dar, die sich auf die Sensorknoten-Hardware beziehen. Abgeschlossen wird das Kapitel durch ein Fazit in Abschnitt 5.7.

5.1 Anlehnung an das BSI-Projekt MoSe

Im Rahmen des BSI-Projekts MoSe wurde ein möglichst generischer Sicherheitsprozess mit Fokus auf Informationssicherheit für Anwendungen drahtloser Sensornetzwerke entwickelt. Es wurden insbesondere eine Vorgehensweise zur Ermittlung des Schutzbedarfs und der Bedrohungsanalyse erarbeitet [18]. Diese basieren sinnvollerweise auf den eingesetzten Anwendungen bzw. den Einsatzszenarien. Der Schutzbedarf ist letztlich abhängig von der konkreten Anwendung, dem Einsatzszenario und dem Auftraggeber.

In diesem Projekt (WSNLAB) wird eine (generische) Experimentierumgebung für WSNs in einem Labor geschaffen. Mithilfe dieser Umgebung ist es möglich, Angriffe gegen ein Sensornetz durchzuführen und Schutzmechanismen zu evaluieren. Da es aber durch den Labor-Charakter weder eine allgemeine Anwendung, noch ein allgemeines Einsatzszenario gibt, erscheint eine Detailanalyse (wie in MoSe vorgeschlagen) nicht uneingeschränkt sinnvoll. Daher wird im Folgenden (in Anlehnung an [18, Kapitel 6]) eine generische Bedrohungsanalyse durchgeführt sowie darauf aufbauend (in Anlehnung an [18, Kapitel 7]) eine generische Sicherheitsarchitektur erstellt.

5.2 Gefahrenanalyse

Die Gefahrenanalyse hat zum Ziel, Gefahren im Sinne von Angriffstypen im Rahmen des betrachteten Netzwerks zu identifizieren. Aufgrund der besonde-

ren Eigenschaften von WSNs (vgl. Kapitel 2) ergeben sich sowohl neuartige Angriffe als auch besondere Herausforderungen in Bezug auf Gegenmaßnahmen: Im Gegensatz zu Knoten in einem Mobile Ad-hoc NETwork (MANET) ist es z.B. durchaus wahrscheinlich, dass sich ein Angreifer direkten physischen Zugang zu einem der Sensorknoten verschafft, da diese für viele Anwendungen einmalig in einem Gebiet verteilt werden und danach frei zugänglich sind [126]. Weiterhin kann keine Ende-zu-Ende Verschlüsselung von Sensordaten vorgenommen werden, wenn In-Network-Processing (vgl. Abschnitt 2.2) erfolgt, da ein Aggregator-Knoten auf die Daten zugreifen können muss [89]. Eine generelle Herausforderung stellt der Mangel an Ressourcen, wie z.B. Prozessorleistung oder Programmspeicher, dar. Dadurch sind weder komplizierte Berechnungen noch das Speichern von großen Datenmengen möglich und Gegenmaßnahmen müssen somit gleichzeitig einfach wie effektiv entwickelt werden.

Bei der Gefahrenanalyse helfen zwei wichtige Merkmale, die Angriffe zu klassifizieren [89]: Zum Einen wird zwischen einem *Outsider-Angriff* und einem *Insider-Angriff* unterschieden. Bei einem Outsider-Angriff wird angenommen, dass der Angreifer keinen direkten Zugriff auf das bestehende Netzwerk hat und lediglich auf das drahtlose Medium zugreifen kann. Bei einem Insider-Angriff hingegen hat der Angreifer legitime Sensorknoten kompromittiert oder Schlüsselmaterial oder ähnliche Daten von legitimen Sensorknoten ausgelesen, um diese Daten zum Eindringen ins Netz zu missbrauchen.

Weiterhin wird zwischen *mote-class* und *laptop-class* Angreifern differenziert [89]. Erstere nutzen Sensorknoten-Hardware, um das WSN anzugreifen, d.h. die Ressourcen des Angreifers sind ebenso beschränkt wie die der legitimen Sensorknoten. Laptop-class Angreifer verfügen über Laptop-ähnliche Hardware und können mit entsprechend vielen Ressourcen aufwarten. Dadurch entsteht ein signifikanter Vorteil, da bspw. ein Jamming-Angriff sowohl länger als auch mit erhöhter Reichweite durchgeführt werden kann.

Die Gefahren werden im Folgenden anhand von vier *Angreifer-Zielen* klassifiziert: Daten können manipuliert (*„Manipulate Data“*), mitgehört (*„Eavesdrop Data“*) oder verworfen (*„Drop Data“*) werden. Dem geht in einigen Fällen (bei Insider-Angriffen) der Zugriff zum Netzwerk (*„Network Access“*) voraus. Diese vier Ziele werden im Folgenden näher beschrieben.

Manipulate Data *Datenmanipulation* kann je nach Anwendungsgebiet des WSNs verheerende Folgen haben. Als Beispiele seien sensorische Patientenüberwachung sowie industrielle Automatisierung an dieser Stelle genannt. Im Wesentlichen kann dieses Ziel auf Routing-Ebene erreicht werden, indem der Angreifer durch Fälschen von Link-Qualitäten Datenverkehr auf sich zieht. Im Kontext von WSNs ist es allerdings auch denkbar, dass der Angreifer Knoten von ihren vorgesehenen Positionen weg bewegt. Da in vielen Anwendungen die Sensorknoten Daten über ihre Umgebung sammeln, sind diese Daten abhängig von der Position. Verschiebt der Angreifer also einen Knoten, so wird fälschlicherweise angenommen, dass die Umgebungsdaten zur ursprünglichen Position des Knotens zuzuord-

nen sind. Das Angreiferziel, Daten zu manipulieren, korrespondiert zur Sicherheitsanforderung *Integrität*.

Eavesdrop Data *Daten mitzuhören* kann dem Angreifer unterschiedliche Vorteile bringen. Zum Einen kann der Angreifer potentiell vertrauliche Daten mithören. Zum Anderen ermöglicht es dem Angreifer die statistische Verkehrsanalyse, wodurch er z.B. bestimmen kann, welche Knoten besonders lohnenswerte Ziele darstellen (auch als „Homing“ bekannt [135,179]). Ferner kann durch Analyse der Paketstruktur herausgefunden werden, welche Protokolle z.B. auf MAC- oder Routing-Ebene verwendet werden. Mit diesem Wissen können effizientere, protokoll-spezifische Angriffe gestartet werden. Speziell in WSNs besteht die Gefahr, dass Arbeits- und Programmspeicher eines Knotens ausgelesen werden. Sind in diesen Schlüssel zum Ver- und Entschlüsseln der Netzwerk-Kommunikation abgelegt, so kann der Angreifer diese nutzen, um sich Zugriff zum WSN zu verschaffen. Das Mithören der Daten korrespondiert zur Sicherheitsanforderung *Vertraulichkeit*.

Drop Data Das *Verwerfen von Daten* ist im weitesten Sinne mit Denial-of-Service (DoS) zu vergleichen und somit eine weitere Gefahr in WSNs. Ein einfacher wie effektiver Angriff ist das Stören des drahtlosen Funksignals. Dadurch müssen sendebereite Knoten, die diesem Angriff lange genug ausgesetzt sind, Datenpakete verwerfen. Der Empfang von Datenpaketen wird durch Interferenz ebenfalls gestört. Eine weitere Realisierung besteht in dem Verwerfen von Daten, die ursprünglich aufgrund von Multi-hop-Kommunikation weitergeleitet werden sollten. Die häufig frei zugänglichen Sensorknoten können ebenso vom Angreifer aus dem Netzwerk entfernt werden. So werden sozusagen alle weiteren Sensordaten dieses Knotens verworfen. Das Verwerfen von Daten korrespondiert zur Sicherheitsanforderung *Verfügbarkeit*.

Network Access Da viele Outsider-Angriffe verhältnismäßig leicht durch Verschlüsselung und Authentifizierung verhindert werden können [89], liegt es im Interesse des Angreifers, *Zugriff zum Netzwerk* zu erlangen. Dies lässt sich z.B. durch Hinzufügen eines Knotens erreichen, indem sich dieser als legitimer Teilnehmer des Netzes ausgibt. Weitaus komplexer gestalten sich hingegen das Auslesen von Schlüsselmaterial aus Arbeits- oder Programmspeicher oder die Übernahme eines legitimen Knotens. Letzteres kann z.B. durch böswilliges Nutzen von OTAP realisiert werden.

Die vier beschriebenen Angreifer-Ziele sind in Abbildung 5.1 in der roten Ebene („Goal“) dargestellt. Diesen Zielen wurden Angriffe zugeordnet (blaue Ebene „Attack“), welche weiterhin anhand der zugehörigen Angriffsebene strukturiert wurden („PHY“, „MAC“, „RTG“ und „NODE“). Der Reihe nach entsprechen diese Ebenen den vier folgenden Abschnitten „Physikalische Ebene“, „Medienzugangs-Ebene“, „Routing-Ebene“ und „Physikalischer Übergriff“. Wie auch in der Abbildung zu erkennen ist, hat diese Strukturierung – im Vergleich zur Unterteilung nach Ziel – den Vorteil, dass die Angriffe eindeutig einer Ebene

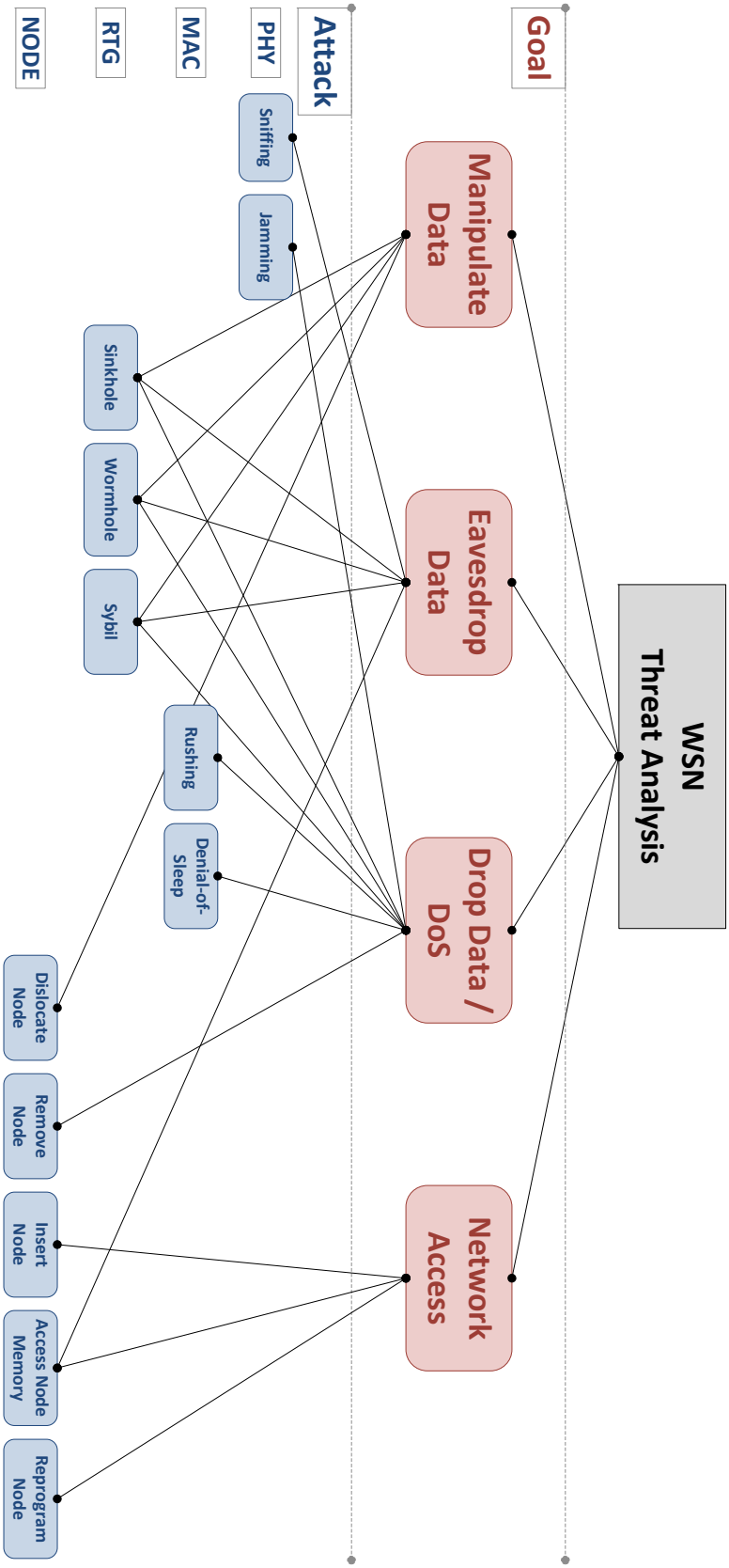


Abbildung 5.1: Gefahrenanalyse für WSNs.

zugewiesen werden können. Im Zuge der Angriffsbeschreibung werden außerdem entsprechende Gegenmaßnahmen vorgestellt. Der Begriff „Gegenmaßnahme“ ist dabei als Oberbegriff für *Präventiv-, Detektions- und Reaktionsmaßnahmen* zu verstehen. Präventivmaßnahmen sind i.A. zu bevorzugen, jedoch nicht immer zulässig. Daher muss in diesen Fällen auf Detektions- und Reaktionsmaßnahmen zurückgegriffen werden. Effektive Reaktionsmaßnahmen sind je nach Angriff allerdings ungleich komplexer und meist auch abhängig vom konkreten Einsatz des Netzwerks. Aus diesem Grund wird i.A. lediglich auf die Detektion des jeweiligen Angriffs eingegangen.

5.3 Physikalische Ebene

Die drahtlose Kommunikation birgt durch das geteilte und frei zugängliche Medium zwei inhärente Risiken. Einerseits kann das passive Mithören eines physikalischen Signals (Sniffing) nicht erkannt werden, andererseits besteht die Gefahr des aktiven Behinderns der Kommunikation durch ein Störsignal (Jamming). Auf beide Risiken und mögliche Gegenmaßnahmen wird im Folgenden eingegangen.

5.3.1 Sniffing

Sniffing beschreibt das Sammeln von Informationen, indem die drahtlose Kommunikation zwischen den Sensorknoten abgehört wird. Wie in [18, Kapitel 3] beschrieben, können dadurch nicht nur die Anwendungsdaten (wie die Messwerte der Sensoren) sondern die gesamte Kommunikation auf der physikalischen Ebene (z.B. auch Routing-Informationen) abgehört werden. Der Prozess des Sniffings ist demnach rein passiv und bedeutet keine Modifikation der Kommunikation. Zudem kann durch Sniffing eine Verkehrsanalyse durchgeführt werden, welche selbst bei verschlüsselter Kommunikation funktioniert, da hier keine Paketinhalte benötigt werden.

Gegenmaßnahmen

Eine offensichtliche Gegenmaßnahme besteht beim Sniffing darin, den gesamten Netzverkehr zu verschlüsseln. Hierbei kann einer Verkehrsanalyse jedoch nicht vorgebeugt werden, da dieser Angriff nicht auf die Inhalte sondern nur auf den Umstand, dass überhaupt Daten übertragen werden, angewiesen ist. Grundsätzlich gibt es verschiedene Ansätze der Verschlüsselung, die in symmetrische und asymmetrische Verfahren unterteilt werden können: Bei *asymmetrischen* Verfahren existieren zwei Schlüssel, einer zum Verschlüsseln und einer zum Entschlüsseln. Bei *symmetrischen* Verfahren hingegen wird ein einziger Schlüssel für beide Vorgänge verwendet. In beiden Fällen stellt sich die Frage, wie die Schlüssel verteilt werden. Bei asymmetrischen Verfahren besitzt

in der Regel jeder Kommunikationspartner ein Schlüsselpaar, wobei die jeweiligen öffentlichen Schlüssel jedem Knoten bekannt sind. Hierdurch ergeben sich $O(n)$ Schlüssel, die jeder der n Knoten kennen muss, um mit allen anderen Knoten kommunizieren zu können. Bei symmetrischen Verfahren ließe sich ein einziger Schlüssel für jegliche Kommunikation verwenden. Hierdurch benötigt jeder Knoten nur einen Schlüssel. Dieser Ansatz hat den Nachteil, dass durch die Kompromittierung eines Knotens die gesamte Kommunikation entschlüsselt werden kann. Um dem entgegenzuwirken, kann für jedes Knotenpaar ein eigener Schlüssel erstellt werden. Von den resultierenden $O(n^2)$ Schlüsseln muss jeder Knoten $O(n)$ Schlüssel kennen. Wird nun ein Knoten kompromittiert, bleibt die Sicherheit der Kommunikation unter den verbleibenden Knoten unberührt. Für weitere Details zum Thema Schlüsselverteilung sei auf [18, Abschnitt 7.4.2] verwiesen.

Da eine asymmetrische Verschlüsselung für jedes Datenpaket zu aufwendig ist, kann diese für die allgemeine Kommunikation im Rahmen des Wireless Sensor Network Lab (WSNLAB)-Projekts nicht verwendet werden. Stattdessen kommt als Gegenmaßnahme nur die symmetrische Verschlüsselung Advanced Encryption Standard (AES) in Frage. Für diese Art der Verschlüsselung existieren Hardware-Implementierungen sowohl im CC2420 Funkchip [159] der TelosB und MicaZ Sensorknoten als auch im Jennic JN5139 Wireless Microcontroller [84] der iSense Hardware. Jede andere Art der Verschlüsselung müsste auf der CPU des Sensorknotens ausgeführt werden, was bei häufiger Nutzung einen hohen Energieverbrauch und somit eine kurze Lebensdauer des gesamten Sensornetzes nach sich ziehen würde. Neben den Energiekosten ist bei einer Software-Implementierung zudem der Speicherplatzbedarf zu berücksichtigen, der einen weiteren limitierenden Faktor darstellt.

Die praktische Anwendbarkeit einer Verschlüsselung des gesamten Datenverkehrs mithilfe der oben genannten Funkchips wird im Rahmen der Umsetzung der Sicherheitsarchitektur im folgenden Kapitel betrachtet. Zu untersuchen sind der Mehrbedarf an Energie sowie der aus der Verschlüsselung resultierende Kommunikationsoverhead.

5.3.2 Jamming

Der *Jamming*-Angriff stellt einen klassischen Outsider-Angriff dar und zielt auf das Verhindern der Verfügbarkeit des Funkmediums ab. Daher kann dieser als DoS-Angriff auf unterster Ebene des OSI-Modells verstanden werden [18]. Ein physikalischer *Jammer* ist dabei eine „Instanz, die gezielt versucht, die physikalische Übertragung und den Empfang der drahtlosen Kommunikation zu behindern“ [180]. Dazu wird eine Störübertragung verwendet, die bewusst nicht konform zum eingesetzten MAC-Protokoll ist und den Funkkanal zu einem beliebigen Zeitpunkt belegt. Das erfolgreiche Belegen des Mediums setzt eine ausreichende Nähe des Jammers zu den Netzwerkteilnehmern voraus, deren Kommunikation durch den Angriff unterbunden werden soll. Dabei kann

der Jammer sowohl durch spezielle Hardware (laptop-class) als auch durch herkömmliche Sensorknoten (mote-class) realisiert werden.

Ein mögliches Ziel dieses Angriffs kann ein teilweises oder vollständiges Unterbinden der Netzwerkkommunikation darstellen, sowie – analog zum Denial-of-Sleep-Angriff (siehe Abschnitt 5.4.2) – das Erzwingen des Verbrauchs von Energiereserven [18, Kapitel 3]. Zudem kann das gezielte Abschirmen einzelner Sensorknoten vom Angreifer beabsichtigt sein [18, Abschnitt 5.4.1.5]. Mittels dieser Abschirmung können Messdaten verfälscht oder unterdrückt werden. In WSNs zur Überwachung (wie z.B. in [59] realisiert) ist so bspw. ein partielles Ausschalten selbiger denkbar.

Es gibt verschiedene Arten eines Jammers, die sich sowohl in ihrer Strategie, ihrer Effizienz als auch ihrem Detektionsaufwand unterscheiden. In [180] werden vier Klassen von Jamming-Angreifern vorgestellt. Der *konstante* Jammer sendet während des Angriffs ein kontinuierliches, physikalisches Störsignal, das sich als erhöhtes Rauschen im Medium auswirkt. Dagegen verwendet der *täuschende* Jammer reguläre Pakete, um aus einem kontinuierlichen Datenstrom ein Störsignal zu bilden, das das Medium dauerhaft belegt. Gewöhnliche Netzwerkteilnehmer glauben, in diesem Datenstrom legitime Übertragungen zu erkennen. Deshalb ist es möglich, dass diese im Empfangs-Modus verharren und während des Angriffs nicht in den Sendezustand wechseln können.

Anstelle des kontinuierlichen Sendens alterniert der *randomisierte* Jammer zufällig zwischen Schlaf- und Jamming-Phasen, in der er sich wie ein konstanter oder täuschender Jammer verhält. In der Schlafphase wird die reguläre Kommunikation dagegen nicht eingeschränkt. Die vierte Klasse umfasst *reaktive* Jammer, deren Ziel nicht das Verhindern des Nachrichtenversands, sondern das Stören des Nachrichtenempfangs ist. Der Vorteil dieser Zielsetzung ist, dass dazu aus Sicht des Angreifers lediglich eine Reaktion auf Kommunikationsversuche notwendig ist und kein dauerhaftes Jamming des Mediums. Dies reduziert den Energieverbrauch und die Detektierbarkeit des Jammers. Das Kommunikationssignal wird gezielt durch ein Störsignal überlagert. Zur Realisierung eines reaktiven Jammers ist jedoch insbesondere in Sensornetzen, in denen die Sendezeit der Pakete typischerweise sehr gering ist, spezielle Hardware erforderlich, die eine schnelle Reaktionszeit ermöglicht.

Gegenmaßnahmen

Mögliche Gegenmaßnahmen gegen einen Jamming-Angriff werden in [18, Abschnitt 7.4.7] diskutiert. Dabei wird zwischen erkennenden und vermeidenden Maßnahmen differenziert, wobei das Alarmieren einer Basisstation und das kollaborative Erkennen durch mehrere Sensorknoten explizit nicht betrachtet wird.

Die *erkennenden* Maßnahmen lassen sich unter dem Begriff Anomalie-Erkennung zusammenfassen und können auf verschiedenen Ebenen realisiert werden. Auf MAC-Ebene ist es je nach eingesetztem MAC-Protokoll möglich, Anomalien in der Carrier Sense Time zu detektieren. Dies ist die Zeit, die

bei einem Sendeversuch für den Zugriff auf das gemeinsame Medium benötigt wird. Auf Anwendungsebene – aber auch auf der Netzwerkebene – kann darüber hinaus die erwartete Paketankunftsrate (Packet Delivery Ratio (PDR)) als Indikator für Jamming-Angriffe dienen. Zudem besteht die Möglichkeit, auf physikalischer Ebene die Stärke empfangener Signale zu überwachen. Dies könnte mithilfe des Received Signal Strength Indicator (RSSI) (siehe [159] und [84]) realisiert werden. Die Anfälligkeit für einen Fehlalarm (false positive) und für nicht erkannte Jamming-Angriffe (false negative) ist bei Anomalie-Erkennung ein Maß für die Güte der Erkennung. Zur Steigerung dieser Güte können die Gegenmaßnahmen verschiedener Ebenen kombiniert werden (Cross-Layer Ansatz).

Als *vermeidende* Gegenmaßnahmen werden in [18, Abschnitt 7.4.7.2.] Frequenzsprung- und Frequenzspreizverfahren aufgeführt. Beide Verfahren versuchen, dem Störsignal des Jammers auf physikalischer Ebene auszuweichen. Während das Verfahren zur Frequenzspreizung weder vom IEEE 802.15.4 Standard unterstützt wird noch mittels aktueller Sensorknoten-Hardware realisierbar ist, ist ein Frequenzsprungverfahren auf diesen Plattformen zumindest theoretisch möglich, wird aber im Rahmen des WSNLAB-Projekts nicht untersucht. Stattdessen wurde sich ausschließlich auf eine Cross-Layer Anomalie-Erkennung beschränkt. Zur Erkennung von Jamming-Angriffen auf Teilnetze ist dabei eine lokale Anomalie-Erkennung in den Sensorknoten erforderlich, da diese Angriffe i.A. nicht alleine durch die Basisstation detektiert werden können.

Das dynamische Routing, das in der Lage ist, den durch den Jammer beeinträchtigten Bereich des Netzwerks zu umgehen, kann als weitere vermeidende Maßnahme aufgefasst werden [18, Abschnitt 7.4.7.2.]. Prinzipiell stellt es aber eher eine notwendige Anforderung an die Robustheit des Routing-Protokolls dar und ist durch beide im Testbed eingesetzten Routing-Algorithmen gewährleistet.

5.4 Medienzugangs-Ebene

Protokolle auf Medienzugangs- (MAC-) Ebene sind für die Koordination des Zugriffs auf den Funkkanal zuständig, um Datenverlust durch Kollisionen zu verhindern und eine faire Nutzung zu gewährleisten. Insbesondere in WSNs kommt außerdem hinzu, dass Mechanismen zur Energiekonservierung auf dieser Ebene umgesetzt werden, da sie hier ohne Bezug zur Anwendung am effektivsten realisiert werden können. Durch all diese Aufgaben bieten MAC-Protokolle eine breite Angriffsfläche, welche durch die im Folgenden vorgestellten Angriffe ausgenutzt werden kann.

5.4.1 Rushing

Die meisten MAC-Protokolle basieren auf der Annahme, dass sich alle Netz-Teilnehmer an dieses halten, da so auch eine gewisse Fairness bzgl. des Medi-

enzugriffs erreicht werden kann. Beim *Rushing*-Angriff wird dieses Vertrauen missbraucht, indem der Angreifer schneller als andere Knoten sendet [179]. Dies lässt sich in den populären CSMA-basierten Protokollen (vgl. auch Abschnitt 2.1.2) durch Ignorieren des Backoff-Timers realisieren. Auf diese Weise kann der Angreifer das Medium für sich beanspruchen, da andere protokollkonforme Knoten den Backoff abwarten.

Rushing resultiert in Unfairness, denn legitime Knoten werden am Senden gehindert. Daher gehört es zur Klasse der DoS-Angriffe. Weiterhin kann Rushing als Hilfsmittel für andere Angriffe verwendet werden. Ein Sinkhole-Angriff (Abschnitt 5.5.1) bspw. setzt das Fälschen von Routing-Informationen voraus. Mit Rushing wird erreicht, dass die gefälschten vor den legitimen Informationen beim Empfänger ankommen.

Falls keine generellen Gegenmaßnahmen gegen Outsider-Angriffe im anzugreifenden Netzwerk getroffen wurden (vgl. Abschnitt 5.2), lässt sich Rushing als solcher durchführen. Meistens wird jedoch von einem Insider-Angriff ausgegangen. Weiterhin benötigt dieser Angriff keine besondere Hardware und kann demnach sowohl laptop-class als auch mote-class Angreifern zugeordnet werden.

Gegenmaßnahmen

Da beim Rushing entgegen den Vorschriften des MAC-Protokolls verfahren wird, bietet sich eine Anomalie-Erkennung zum Detektieren an. Genauer muss eine Anomalie der Carrier Sense Time erkannt werden (vgl. auch Abschnitt 5.3.2). Der Angreifer gibt sich durch eine sehr kurze Carrier Sense Time zu erkennen, da legitime Knoten den Backoff Timer beachten. Da es sich um einen Angriff auf MAC-Ebene handelt, muss die Anomalie-Erkennung lokal auf den Knoten erfolgen, die sich in der direkten Nachbarschaft des Angreifers befinden.

5.4.2 Denial-of-Sleep

Wie in Abschnitt 2.1.1 bereits ausführlich diskutiert, ist im Bereich der WSNs Energieeffizienz der wichtigste Faktor beim Software-Design im Allgemeinen und Protokoll-Design im Besonderen. Daher stellt die Ressource Energie eines der lukrativsten Angriffsziele dar. Die meiste Energie wird durch Kommunikation und somit durch den Funkchip aufgewendet. Da das MAC-Protokoll die Steuerung, d.h. das An- und Ausschalten, des Funkchips übernimmt, kann auf dieser Ebene am meisten bewirkt werden. Der sogenannte *Denial-of-Sleep*-Angriff [134, 135], auch bekannt als „Battery Exhaustion“ oder „Sleep Deprivation“, nutzt dies aus und ist eine spezielle Form von DoS-Angriff. Die Grundidee dabei ist, den Funkchip davon abzuhalten, in den Sleep-Modus zu gehen und Energie zu sparen.

Denial-of-Sleep kann zwar auch durch Jamming erreicht werden (vgl. auch Abschnitt 5.3.2 und [18, Kapitel 3]), ist aber auf physikalischer Ebene bei weitem

nicht so effektiv wie auf MAC-Ebene [134]. Intelligente Denial-of-Sleep-Angriffe können die Lebensdauer eines WSNs von mehreren Monaten auf wenige Tage reduzieren. Dazu werden Informationen über das verwendete MAC-Protokoll benötigt, was allerdings relativ leicht durch Verkehrsanalyse erreicht werden kann [135].

Es gibt mehrere Möglichkeiten, das MAC-Protokoll dazu zu bewegen, den Funkchip aktiv zu halten und auf diese Weise Energie zu verschwenden. Die vier wesentlichen sind dabei die folgenden [134]:

Kollision Kollidiert ein gesendetes Paket mit einer anderen Übertragung von ausreichend hoher Sendestärke, so werden einzelne Bits in diesem Paket korrumpiert und das komplette Paket muss neu gesendet werden. Forward Error Corrections (FECs) können dieses Problem in manchen Fällen lösen, stehen aber im Gegensatz zur Optimierung der Übertragungszeit. Kollisionen können ebenso auf physikalischer Ebene durch Jamming hervorgerufen werden, da kein gültiges Datenpaket erforderlich ist.

Kontrollpaket-Overhead Kontrollpakete auf MAC-Ebene müssen in der Regel von allen direkten Nachbarn des Senders empfangen werden. Fälscht der Angreifer diese Kontrollpakete, können so direkt mehrere Knoten gleichzeitig angegriffen werden.

Overhearing Overhearing entsteht, wenn der Funkchip eines Knotens sich im Empfangs-Modus befindet, das auf dem Medium übertragene Paket jedoch für einen anderen Knoten bestimmt ist. Wie in Abschnitt 2.1.2 beschrieben, nutzen gängige MAC-Protokolle für WSNs Duty-Cycling-Mechanismen, um Overhearing zu minimieren. Eben diese Mechanismen lohnen sich besonders als Angriffspunkt bei einem Denial-of-Sleep-Angriff und müssen in Abhängigkeit vom jeweiligen Protokoll ausgenutzt werden.

Idle-Listening Da der Energieverbrauch im Idle-Modus und Empfangs-Modus identisch ist, ist ein Denial-of-Sleep-Angriff genauso effektiv, wenn sich das angegriffene Ziel nur im Empfangs-Modus befindet.

Raymond et al. [134] haben unterschiedliche Klassen von Denial-of-Sleep-Angriffen untersucht. Dabei sind Insider-Angriffe sowie Angreifer mit Kenntnis über das eingesetzte MAC-Protokoll deutlich effektiver als Outsider-Angriffe oder wenn das MAC-Protokoll nicht bekannt ist. Außerdem wirken sich Outsider-Angriffe mit Kenntnis über das MAC-Protokoll sehr viel stärker auf das Netzwerk aus als konstantes Jamming. Weiterhin wurden Schwachstellen in unterschiedlichen MAC-Protokollen analysiert mit dem Ergebnis, dass die verschiedenen Energiekonservierungs-Mechanismen unterschiedlich stark anfällig sind. Allerdings konnte für jedes Protokoll eine Variation gefunden werden, so dass der Angreifer nicht mehr Energie aufwenden muss als diese beim Opfer verschwendet wird. Daraus folgt, dass ein laptop-class Angreifer zwar mehr Knoten angreifen kann, aber auch ein mote-class Angreifer ausreicht, um die Batterie eines einzelnen Zielknotens zu entleeren.

Gegenmaßnahmen

Zusätzlich zur Angriffs-Analyse wird in [134] auch ein umfangreiches Framework zur Abwehr von Denial-of-Sleep-Angriffen vorgestellt. Dieses besteht aus vier einzelnen Komponenten, die jeweils eine der identifizierten Sicherheitslücken abdecken sollen: Eine starke Authentifizierung auf Link-Ebene soll verhindern, dass der Angreifer gesicherten Datenverkehr auf MAC-Ebene versenden kann. Weiterhin verhindert Schutz vor Replay-Angriffen das unnötige Versenden von altem Datenverkehr. Die dritte Komponente besteht aus Gegenmaßnahmen zu Jamming (vgl. auch Abschnitt 5.3.2). Ein leichtgewichtiger Intrusion-Detection Mechanismus soll zudem unauthentifizierte Broadcast-Angriffe erkennen. Aufgrund der hohen Komplexität und des hohen Implementierungsaufwands wurde das Framework innerhalb dieses Projekts nicht weiter verfolgt. Stattdessen ist die Umsetzung einer Anomalie-Erkennung bzgl. des Energieverbrauchs zur Detektion von Denial-of-Sleep-Angriffen denkbar. Da ein Denial-of-Sleep-Angriff einen ungewöhnlich hohen Energieverbrauch bewirkt, kann dieser vom gewöhnlichen Verbrauch unterschieden und somit detektiert werden. Dazu muss zunächst statistisch ermittelt werden, wie „gewöhnlich“ quantitativ ausgedrückt werden kann.

5.5 Routing-Ebene

Routing-Protokolle nehmen ähnlich wie MAC-Protokolle an, dass Knoten kooperieren und reale Angaben z.B. über ihre Nachbarschaft oder Link-Qualitäten machen. Angriffe auf Routing-Ebene haben gemeinsam, dass sie dies ausnutzen und Informationen über die Netzwerk-Topologie auf bestimmte Weise verändern. Dazu wird in der Regel das Fälschen von Routing-Paketen bzw. Routing-Informationen vorausgesetzt, wobei eine Kombination mit Rushing (vgl. Abschnitt 5.4.1) von Vorteil sein kann.

5.5.1 Sinkhole

Zu den mächtigsten Routing-Angriffen zählt der *Sinkhole*-Angriff, der zum Ziel hat, maximalen Datenverkehr zum oder über den Angreifer zu locken [89]. Da auf diese Weise die meiste Kommunikation über den Angreifer läuft, kann dieser auch entscheiden, was mit den Datenpaketen passiert.

Entgegen der Beschreibung in [18, Kapitel 3] ist dieser Angriff i.A. nicht nur darauf beschränkt, die Identität der Senke anzunehmen oder überhaupt eine Identität zu fälschen. Stattdessen erfordert ein Sinkhole-Angriff, dass sich der Angreifer besonders attraktiv in Bezug auf die Routing-Metrik darstellt. Da die meisten Routing-Protokolle optimale Pfade (bzgl. der Routing-Metrik) verwenden, wird der Angreifer so von seinen Nachbarn als Relay bevorzugt. Diese attraktive Route wird außerdem zu weiter entfernten Knoten propagiert, so dass auch diese den Angreifer als Relay auf dem Pfad zum Empfänger nutzen.

Aufgrund des besonderen Kommunikationsmusters in WSNs (Convergecasts, vgl. Abschnitt 2.2.1) ist es als Angreifer dabei sinnvoll, sich in Nähe der Senke (Basisstation) zu positionieren.

Es existieren mehrere Methoden, ein Sinkhole zu realisieren, wobei in der Regel von einem Insider-Angriff ausgegangen wird. Einerseits können die Qualitäten der eigenen Links zu Nachbarknoten in Routing-Paketen gefälscht werden. Andererseits kann sogenanntes ACK Spoofing auf Link-Ebene (vgl. z.B. [89]) verwendet werden. Hierbei wird ausgenutzt, dass sich viele Routing-Protokolle zur Bestimmung der Link-Qualität auf ACKs auf Link-Ebene verlassen. Der Angreifer kann die Identität eines Knotens annehmen, dessen Link schwach oder ausgefallen ist, und ein entsprechendes ACK fälschen. Abgesehen von Methoden, wo eine vorteilhafte Route zur Senke vorgetäuscht wird, gibt es auch Methoden, bei denen diese wirklich vorliegt, aber nicht von den legitimen Knoten gesehen werden kann: Wird das Sinkhole von einem laptop-class Angreifer eingesetzt, so kann er mit einer höheren Funkreichweite die Basisstation über nur einen Hop erreichen, obwohl er weiter von ihr entfernt ist als andere Sensorknoten.

Ein Sinkhole ermöglicht es, verschiedene weitere Maßnahmen durchzuführen. Diese werden i.A. als *Blackhole* und *Greyhole* bezeichnet. Bei einem Blackhole-Angriff werden alle Pakete, die eigentlich zur Weiterleitung bestimmt sind, verworfen. Bei einem Greyhole-Angriff werden Datenpakete hingegen nur selektiv verworfen bzw. weitergeleitet (auch bekannt als „Selective Forwarding“). Daher kann ein Greyhole auch als Verallgemeinerung eines Blackholes angesehen werden. Ein Greyhole-Angriff ist dabei deutlich schwerer zu erkennen, da sporadische Paketverluste in einem drahtlosen Netzwerk nicht ungewöhnlich sind. Weiterhin ist es natürlich auch möglich, Daten zu analysieren oder sogar zu modifizieren, bevor sie weitergeleitet werden.

Gegenmaßnahmen

Sinkhole-Angriffe sind i.A. schwierig abzuwehren [89]. Das hängt vor allem damit zusammen, dass die Angaben zu Routing-Metriken (sogenannte „Advertisements“), wie z.B. Ende-zu-Ende Delay oder Zustellverlässlichkeit zur Basisstation, schwer zu verifizieren sind. Aufgrund seiner Mächtigkeit verlangt der Sinkhole-Angriff nach zuverlässigen und für WSNs angepasste Gegenmaßnahmen. Im Rahmen dieses Projekts wurde zunächst eine heuristik-basierte, protokoll-abhängige Detektion angestrebt.

5.5.2 Wormhole

Ein *Wormhole*-Angriff ist eine spezielle Art eines Replay-Angriffs, bei dem Datenpakete von einem zum anderen Ende des Netzwerks geschleust werden [89] (vgl. auch [18, Kapitel 3 und Abschnitt 7.4.6.2]). Aufgrund der engen Verwandschaft zum Sinkhole-Angriff ist der Wormhole-Angriff als ebenso mächtig und gefährlich einzustufen (siehe daher auch Abschnitt 5.5.1). Mittels einer schnel-

len out-of-band Verbindung, die nur dem Angreifer zur Verfügung steht, werden Pakete, die an einem Ende des sogenannten Wormhole-Links empfangen wurden, an das andere Ende übertragen und dort wieder eingespielt. Die out-of-band Verbindung kann dabei bspw. kabelgebunden sein und verfügt über eine hohe Bandbreite, sodass für legitime Knoten am anderen Ende des Wormholes der Eindruck entsteht, dass der ursprüngliche Sender des wieder eingespielten Pakets nur wenige Hops entfernt liegt.

Ein simpler Wormhole-Angriff kann bereits darin bestehen, dass der Angreifer zwischen zwei Knoten liegt und Pakete zwischen diesen weiterleitet (wobei der Angreifer nicht notwendigerweise als Relay bestimmt war). Im Allgemeinen geht man aber davon aus, dass der Angreifer über zwei Knoten verfügt, welche über die out-of-band Verbindung kommunizieren. Weiterhin ist es nicht notwendig für den Angreifer, Zugang zum Netzwerk zu haben, denn für diesen speziellen Replay-Angriff reicht es aus, die Daten mitzuschneiden und sie so wieder einzuspielen wie sie empfangen wurden.

Im Kontext von WSNs ist ein Wormhole am effektivsten, wenn ein Ende des selbigen in Nähe der Basisstation liegt und das andere Ende mehrere Hops davon entfernt. Damit kann den weiter entfernten Knoten mithilfe des Wormholes vorgetäuscht werden, in Nähe der Basisstation zu sein. Die zum entfernten Wormhole-Ende benachbarten Knoten werden außerdem das Wormhole als Pfad zur Senke wählen, da es viel attraktiver ist als andere legitime, benachbarte Knoten, welche mehrere Hops zu überwinden haben. Auf diese Weise kann also ebenfalls ein Sinkhole realisiert werden und der potentiell gesamte Datenverkehr wird über das Wormhole zur Basisstation geschleust.

An dieser Stelle erwähnenswert ist das *HELLO Flooding* [89] als Spezialfall eines in-band Wormhole-Angriffs. Hierbei handelt es sich um das Wiedereinspielen von HELLO-Paketen. Empfängt ein Knoten ein solches Routing-Kontrollpaket, kann er annehmen, dass der Sender in direkter Funkreichweite liegt. Ein laptop-class Angreifer kann dies ausnutzen, indem er seine große Reichweite einsetzt, um HELLO-Pakete in mehrere Hops entfernten Teilen des Netzwerks wieder einzuspielen. So entsteht für dort positionierte Knoten der Eindruck, dass sie in direkter Nachbarschaft zum Angreifer stehen, können diesen aber eigentlich gar nicht erreichen. Kündigt der Angreifer auf diese Weise auch noch eine optimale Route zur Basisstation an, so werden alle protokoll-konformen Knoten den Angreifer als Relay wählen.

Gegenmaßnahmen

Wormhole-Angriffe sind ebenso wie Sinkhole-Angriffe schwer zu verhindern oder zu erkennen [89]. Dies hat im Wesentlichen mit der out-of-band Verbindung zu tun, die nur dem Angreifer bekannt und für die legitimen Knoten unsichtbar ist. Ansätze, die wie Packet Leashes [79] eine strikte Zeitsynchronisation voraussetzen, sind im Rahmen von WSNs i.A. nicht realisierbar. Analog zum Sinkhole-Angriff wurde daher zunächst ein heuristik-basierter Ansatz verfolgt.

5.5.3 Sybil

Bei einem *Sybil*-Angriff nimmt der Angreifer gleich mehrere Identitäten an [89] (vgl. auch [18, Kapitel 3 und Abschnitt 7.4.6.2]). Dies kann bspw. von Vorteil beim Einsatz von Multi-Path-Routing-Protokollen sein, die mehrere Pfade für das Routing nutzen, anstatt sich auf einen festzulegen. So wird (fälschlicherweise) angenommen, dass Pfade mit disjunkten Knoten verwendet werden, in Wahrheit handelt es sich beim Angreifer aber um ein und denselben Knoten mit multiplen Identitäten. Bei diesen Multi-Path-Routing-Protokollen ist ein Sybil-Angriff sinnvoll einsetzbar als Vorbereitung zum Sinkhole-Angriff (vgl. Abschnitt 5.5.1). Weiterhin ist der Sybil-Angriff auch ein Sicherheitsrisiko für Lokalisierungs-Protokolle [18, Abschnitt 7.4.6.2].

Gegenmaßnahmen

Generell kann ein Sybil-Angriff verhindert werden, indem Identitäten durch Authentifizierungsmechanismen verifiziert werden. Dies kann z.B. mithilfe einer Trusted Sensor Node (TSN) Basisstation realisiert werden. Da die Hauptgefahr dieses Angriffs jedoch in der Vorbereitung eines Sinkholes liegt, wird dieser nicht explizit behandelt. Stattdessen sei an dieser Stelle auf Gegenmaßnahmen zum Sinkhole-Angriff (Abschnitt 5.5.1) verwiesen.

5.6 Physikalischer Übergriff

Die Klasse der physikalischen Übergriffe enthält verschiedene, WSN-spezifische Angriffe, die unabhängig von einer konkreten Protokoll-Ebene sind. Auf der einen Seite werden Sensorknoten häufig in Umgebungen benutzt, in der sie ungeschützt und für potentielle Angreifer physisch zugänglich sind. Es besteht die Gefahr, dass Sensorknoten bewegt, entwendet oder manipuliert werden. Wird die physische Erreichbarkeit der Knoten erschwert, kann auf diese Weise die Wahrscheinlichkeit für einen derartigen Angriff verringert werden [18, Kapitel 4]. Auf der anderen Seite entsteht in Anwendungen, die eine Reprogrammierung der Sensorknoten über die Funkschnittstelle erlauben, eine mögliche Angriffsfläche für physikalische Übergriffe. Im Folgenden wird genauer auf diese WSN-spezifischen Angriffe eingegangen.

5.6.1 Dislocate Node

Das gezielte Verändern der Position (*Dislocation*) von Sensorknoten, die ungeschützt und unbefestigt ausgebracht wurden, ist ein leicht zu realisierender Outsider-Angriff. Der Angreifer verfolgt dabei in der Regel das Ziel, die Lokalisierungsanwendung zu manipulieren und dadurch Messdaten zu verfälschen. Dies gilt insbesondere dann, wenn die Position von Ankerknoten mit fest einkodierter Position unbemerkt geändert wird (vgl. [18, Abschnitt 7.4.6.2]).

Gegenmaßnahmen

In statischen WSNs kann eine unautorisierte Positionsveränderung mithilfe eines Beschleunigungssensor erkannt werden. In mobilen WSNs ist es in der Regel nicht möglich, unautorisierte Positionsveränderungen von Sensorknoten zu bemerken. Daher beschränken sich die Gegenmaßnahmen zur Positionsveränderung auf eine Detektion des Angriffs in Anwendungen mit statischer Platzierung der Sensorknoten. Es ist denkbar, diese Gegenmaßnahmen einerseits mit Hilfe der Beschleunigungssensorik, andererseits über eine Lokalisierungsanwendung zu erreichen, die in der Lage ist, Manipulationen der Knotenpositionierung zu detektieren.

5.6.2 Remove Node

Neben dem räumlichen Entfernen von Sensorknoten muss in einer ungeschützten Umgebung der Verlust von Sensorknoten einkalkuliert werden. Durch äußere Einwirkungen ist es möglich, dass Sensorknoten gewaltsam zerstört oder in ihrer Kommunikationsfunktion beeinträchtigt werden (vgl. [18, Kapitel 3 und Abschnitt 5.4.1.5]). Ein Angriff mit der Absicht, anhand äußerer Einwirkungen einen Knoten aus dem Netzwerk zu entfernen (*Remove Node*), dient primär dem Einschränken der Gesamtfunktion des Sensornetzes und wird folglich der Klasse der DoS-Outsider-Angriffe zugeordnet. Ein weiteres Ziel eines solchen Angriffs kann analog zur Positionsveränderung von Sensorknoten eine Manipulation der Lokalisierungsanwendung (vgl. [18, Abschnitt 7.4.6.2]) oder schlichtweg die finanzielle Schädigung des Betreibers darstellen.

Gegenmaßnahmen

Gegenmaßnahmen zur Ausfallerkennung von Sensorknoten werden in [18, Abschnitt 7.4.9] beschrieben. Dabei wird das Fazit gezogen, dass kaum verfügbare Protokolle für einen derartigen Detektionsmechanismus existieren, der sich jedoch generell in jede Anwendung integrieren lässt und vom Entwickler durchzuführen ist (siehe [18, Abschnitt 7.4.9.3]). Aus diesem Grund wurde die Entwicklung eines zentralisierten, auf periodischen Statusnachrichten (*Heartbeats*) basierenden Verfahrens zur Ausfallerkennung im Testbed vorgesehen. In Abhängigkeit der im Testbed laufenden Anwendung ist zudem nach Möglichkeit beabsichtigt, die Heartbeats in den Sensordatenverkehr zu integrieren, um den zusätzlichen Overhead dieser Gegenmaßnahme zu reduzieren.

5.6.3 Insert Node

Dieser Angriff beschreibt das Hinzufügen eines fremden Knotens in ein bestehendes Sensornetz (vgl. [18, Kapitel 3]). Der Knoten kann dabei sowohl laptop- als auch mote-class sein. Das Ziel des Angriffs ist, durch den Netzwerkzugang

Insider-Angriffe auszuführen. Im Vergleich zur Kompromittierung eines legitimen Knotens (vgl. Abschnitt 5.6.5) besteht hier der Vorteil darin, einen laptop-class Knoten für effektivere Folgeangriffe hinzufügen zu können.

Gegenmaßnahmen

Voraussetzung für das erfolgreiche Hinzufügen eines Sensorknotens ist, die Kommunikation des bestehenden Sensornetzes empfangen und verarbeiten zu können. Diese Voraussetzung kann bereits verhindert oder zumindest erheblich erschwert werden, indem die gesamte Kommunikation mit AES (vgl. Abschnitt 5.3.1) verschlüsselt wird. Dabei kann der verwendete Schlüssel als Authentifizierungsmerkmal angesehen werden.

5.6.4 Access Node Memory

Bei diesem physikalischen Übergriff wird davon ausgegangen, dass ein Angreifer Zugriff auf die Speichermodule eines Sensorknotens erlangt hat und diese auslesen kann (vgl. [18, Abschnitt 5.4.1.5]). Für diesen Fall ist es irrelevant, ob der betreffende Sensorknoten noch aktiv im Sensornetz ist oder nicht (siehe auch Abschnitt 5.6.1 und 5.6.2). Bei diesem Angriff wird außerdem von einem laptop-class Angreifer ausgegangen, der über entsprechende Schnittstellen und Software verfügt, um die Daten auszulesen. Ein häufiges Ziel ist, an Schlüsselmaterial auf diesem Weg zu gelangen, um einen fremden Knoten in das verschlüsselte Netzwerk einzuschleusen (vgl. Abschnitt 5.6.3).

Gegenmaßnahmen

Um das Auslesen von Informationen auf den Speichermodulen zu verhindern, kann der Speicher verschlüsselt werden. Verschiedene Ansätze werden in [18, Kapitel 3] aufgezeigt. Wie schon in Abschnitt 5.3.1 beschrieben, kommt als Verschlüsselungsmethode nur die symmetrische Verschlüsselung nach dem AES Verfahren in Frage. Diese Art der Verschlüsselung wird durch den Funkchip CC2420 ermöglicht. Er bietet neben der Verschlüsselung des Datenverkehrs die Möglichkeit, zu gegebenem Klartext den entsprechenden Chiffretext zurückzuliefern [159]. So kann jede Information, die gespeichert wird, vorher verschlüsselt werden. Hierbei besteht jedoch die Herausforderung, den entsprechenden Schlüssel geeignet abzulegen. Eine Speicherung auf dem Speichermodul in Klartext ist ausgeschlossen, da hierdurch der ursprüngliche Sinn der Verschlüsselung der restlichen Daten in Frage gestellt wird. Den Schlüssel permanent im Arbeitsspeicher zu halten, ist im Allgemeinen auch nicht sinnvoll und je nach Anwendung und Einsatz nicht realisierbar. Insgesamt besteht in diesem Punkt somit noch weiterer Forschungsbedarf.

5.6.5 Reprogram Node

Der Reprogram Node Angriff bezeichnet das gezielte Reprogrammieren eines Sensorknotens – also das Ersetzen des auf dem Knoten vorhandenen Codes durch beliebigen Code – und dient dem Einschleusen von Schadsoftware in das Sensornetz [18, Kapitel 3]. Zur Durchführung eines *Reprogramming*-Angriffs existieren zwei Alternativen. Auf der einen Seite besitzt der Angreifer durch den physischen Zugang auf Sensorknoten in einer ungeschützten Umgebung potentiellen Zugang auf den Programmspeicher (siehe [18, Abschnitt 5.4.1.5]) und damit die Möglichkeit, diesen auszulesen und auch zu überschreiben. Auf der anderen Seite besteht in Sensornetzen, in denen eine OTAP-Anwendung vorgesehen ist, die Gefahr eines unautorisierten Gebrauchs dieser Anwendung mit dem Ziel, schädlichen Code einzuschleusen. Bei dieser Alternative handelt es sich um einen Insider-Angriff, da zur Ausführung direkter Netzzugang erforderlich ist, während es sich i.A. bei der ersten Alternative um einen laptop-class Outsider-Angriff handelt.

Gegenmaßnahmen

Sensorknoten, die einem Reprogramming-Angriff zum Opfer gefallen sind, werden als *kompromittierte* Knoten bezeichnet. Die Schwierigkeit bei der Detektion kompromittierter Knoten ist, dass sich diese auf unbestimmte Zeit völlig protokoll-konform verhalten können. Aus diesem Grund ist es i.A. unmöglich, diese Knoten zu entdecken [18, Kapitel 3] und es bedarf präventiver Maßnahmen gegen das Kompromittieren.

Das Reprogrammieren von Sensorknoten über eine Hardware-Schnittstelle mit gleichzeitigem Auslesen des Programmspeichers erfordert einen hohen Aufwand und ist aus Sicht des Angreifers relativ teuer. Daher ist diese Alternative einem Angreifer nicht in unbeschränktem Umfang möglich [18, Abschnitt 5.4.1.6]. Schutzmechanismen gegen diese Variante sind auf Hardware-Ebene zu realisieren und werden in diesem Projekt nicht untersucht.

Zur Prävention des Reprogramming-Angriffs, der auf eine bestehende OTAP-Anwendung abzielt, gilt zunächst die Empfehlung des im MoSe-Abschlussbericht spezifizierten Basisschutzes: „Doch gerade die Möglichkeit einer Rekonfiguration und insbesondere die Reprogrammierung des Sensornetzes im laufenden Betrieb bietet einem Angreifer u.U. eine große Angriffsfläche. Die Rekonfigurierbarkeit eines Sensornetzes sollte deswegen wenn möglich gemieden werden.“ [18, Kapitel 4, S. 32].

Ist auf die Reprogrammierung des Sensornetzes im laufenden Betrieb dagegen nicht verzichtbar, muss zur Prävention des Reprogramming-Angriffs die Autorisierung der Reprogrammierung sichergestellt werden. Um die bestehenden OTAP-Anwendungen vor unautorisiertem Einspielen bösartiger Software-Aktualisierungen zu schützen, ist es daher vorgesehen, diese durch Kryptographie zu einem signatur-basierten OTAP zu erweitern. Zur Authentifizierung

des Initiators der Reprogrammierung mittels Signaturen sind sowohl symmetrische Verfahren (AES als Cipher Block Chaining Message Authentication Code (CBC-MAC)) als auch asymmetrische Verfahren denkbar.

Die Verwendung symmetrischer Kryptographie zur Authentifizierung durch einen einzelnen, netzwerkweiten Schlüssel, birgt das Risiko, dass ein Angreifer durch Kompromittierung eines Sensorknotens an diesen Schlüssel gelangt und in der Lage ist einen OTAP-Prozess zu initiieren. Wird dagegen zwischen dem autorisierten OTAP-Initiator und jedem Sensorknoten paarweise ein eindeutiger Schlüssel vergeben, wird dieses Risiko vermieden. Die aufwendigere paarweise Schlüsselvergabe besitzt jedoch den Nachteil, dass in Anwendungsfällen, in denen eine Gruppe von Sensorknoten oder gar das gesamte Netzwerk aktualisiert werden soll, der Programm-Code separat an jeden Empfänger übertragen werden muss.

Beim Einsatz asymmetrischer Authentifizierungsmechanismen, welche die Gefahr von unautorisierten OTAP-Prozessen aufgrund von Knotenkompromittierungen ebenfalls minimieren, – da zur Authentifizierung benötigte (private) Schlüssel ausschließlich autorisierten OTAP-Initiatoren bekannt sind, – bleibt dagegen ein effizientes gruppen- bzw. netzwerkweites OTAP gewährleistet. Dieser signifikante Vorteil asymmetrischer Verfahren macht deren Einsatz für das sichere OTAP in WSNs trotz des höheren Ressourcenbedarfs (vgl. Abschnitt 5.3.1) erstrebenswert. Die geringere Laufzeit-Performance asymmetrischer Kryptographieoperationen, die Hardware-seitig nicht unterstützt werden und in Software zu implementieren sind, ist für OTAP-Prozesse als unkritisch einzustufen.

5.7 Fazit

In diesem Kapitel wurde zunächst eine Gefahrenanalyse für WSNs durchgeführt. Dabei wurden die wesentlichen Ziele des Angreifers identifiziert und erläutert. Darauf wurden diesen Zielen Angriffe zugeordnet, wobei ein Angriff möglicherweise mehrere dieser Ziele haben kann. Aus diesem Grund wurde für die darauffolgende detaillierte Beschreibung der Angriffe eine Strukturierung anhand von Angriffsebenen gewählt, die eine eindeutige Zuordnung zulässt. Neben der Angriffsbeschreibung wurden außerdem zu jedem Angriff Gegenmaßnahmen vorgestellt. Das Gefahrenanalyse-Diagramm aus Abbildung 5.1 wurde in Abbildung 5.2 um diese Gegenmaßnahmen (grüne Ebene „Countermeasure“) ergänzt. Zusammengefasst sind dies die folgenden Gegenmaßnahmen:

Encrypt Traffic Das Verschlüsseln des Datenverkehrs ist eine Präventivmaßnahme gegen die beiden Outsider-Angriffe Sniffing und Insert Node (vgl. Abschnitt 5.3.1 bzw. 5.6.3).

Protocol-specific Detection Die protokoll-spezifische Detektion bezieht sich auf die Routing-Angriffe Sinkhole, Wormhole und Sybil (vgl. Ab-

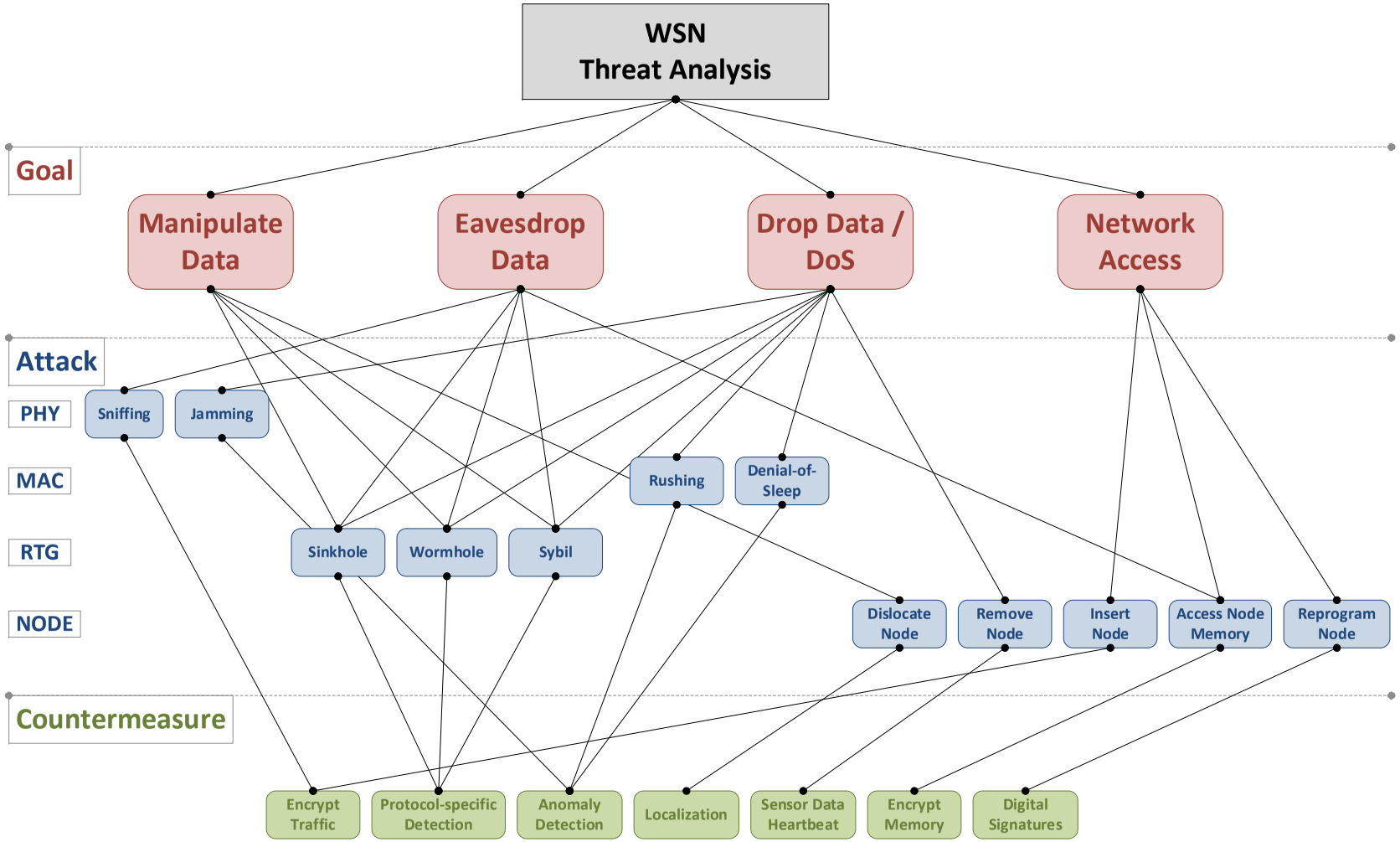


Abbildung 5.2: Gefahrenanalyse für WSNs mit Gegenmaßnahmen.

schnitt 5.5). Genauer werden heuristik-basierte Ansätze zur Detektion angestrebt.

Anomaly Detection Anomalie-Erkennung kann zur Detektion der beiden Insider-Angriffe Rushing und Denial-of-Sleep auf MAC-Ebene (vgl. Abschnitt 5.4) sowie von Jamming (vgl. Abschnitt 5.3.2) angewendet werden.

Localization Lokalisierung ist eine Präventivmaßnahme gegen den Outsider-Angriff Dislocate Node (vgl. Abschnitt 5.6.1).

Sensor Data Heartbeat Zum Detektieren des Outsider-Angriffs Remove Node (vgl. Abschnitt 5.6.2) werden sogenannte Sensordaten-Heartbeats eingesetzt.

Encrypt Memory Den Sensorknoten-Speicher zu verschlüsseln ist eine Präventivmaßnahme gegen den Outsider-Angriff Access Node Memory (vgl. Abschnitt 5.6.4).

Digital Signatures Digitale Signaturen sind als Präventivmaßnahme gegen den Reprogram Node Angriff (vgl. Abschnitt 5.6.5) vorgesehen.

Diese sieben Gegenmaßnahmen decken alle identifizierten Angriffe ab und stellen somit die gesamte Sicherheitsarchitektur dar.

6 Umsetzung der Sicherheitsarchitektur

Dieses Kapitel beschreibt die praktische Umsetzung des im vorherigen Kapitel erstellten Konzeptes der Sicherheitsarchitektur. Zunächst erfolgt die Beschreibung des Intrusion Detection System (IDS) (Abschnitt 6.1), welches die zentrale Anlaufstelle in der Sicherheitsarchitektur darstellt. Aufgrund der Ressourcenbeschränktheit der Sensorknoten, insbesondere bzgl. des Speichers, können jedoch nicht alle Angriffe und Gegenmaßnahmen in einem Sensorknoten-Image untergebracht werden. Daher wurden ausgewählte Angriffe und deren Gegenmaßnahmen möglichst effizient und optimiert realisiert. Als erstes wurde der Sinkhole-Angriff als repräsentativer Routing-Angriff betrachtet (Abschnitt 6.2). Weiterhin wurde ein Ansatz zur Anomalieerkennung der Carrier-Sense-Time (CST) implementiert (Abschnitt 6.3). Details zur Implementierung und Evaluation verschiedener kryptographischer Verfahren sind in Abschnitt 6.4 zu finden. Zudem stellt das sichere OTAP (Secure OTAP (SecOTAP)) eine wichtige Komponente der Architektur dar (Abschnitt 6.5). Im Anschluss an die Beschreibung diverser Komponenten der Sicherheitsarchitektur wird die Skalierbarkeit der Gesamtarchitektur simulativ evaluiert (Abschnitt 6.6). Darauf werden egoistische Sensorknoten betrachtet und Gegenmaßnahmen beschrieben (Abschnitt 6.7). Schließlich wird das Kapitel durch ein Fazit in Abschnitt 6.8 abgeschlossen.

6.1 Intrusion Detection System

Das Intrusion Detection System (IDS) bildet die Kernkomponente der im Rahmen dieses Projektes umgesetzten Sicherheitsarchitektur und dient der Überwachung des Sensornetzes. Dem Betreiber eines WSNs bietet das IDS Informationen über den Netzstatus. Dies beinhaltet die Alarmierung, im Falle detektierter Angriffe innerhalb des Sensornetzes. Dabei handelt es sich um die im Konzept der Sicherheitsarchitektur erwähnten Angriffe (siehe Abbildung 5.2 sowie Kapitel 5). Zudem ermöglicht das IDS die Protokollierung aller Status- und Alarmnachrichten.

In diesem Abschnitt wird die Umsetzung des IDS im Rahmen der Sicherheitsarchitektur behandelt. Dazu wird im Folgenden zunächst ein Überblick über das Konzept und die Architektur des IDS gegeben. Anschließend wird auf die Funktionalitäten des IDS-Servers und auf diverse IDS-Module eingegangen.

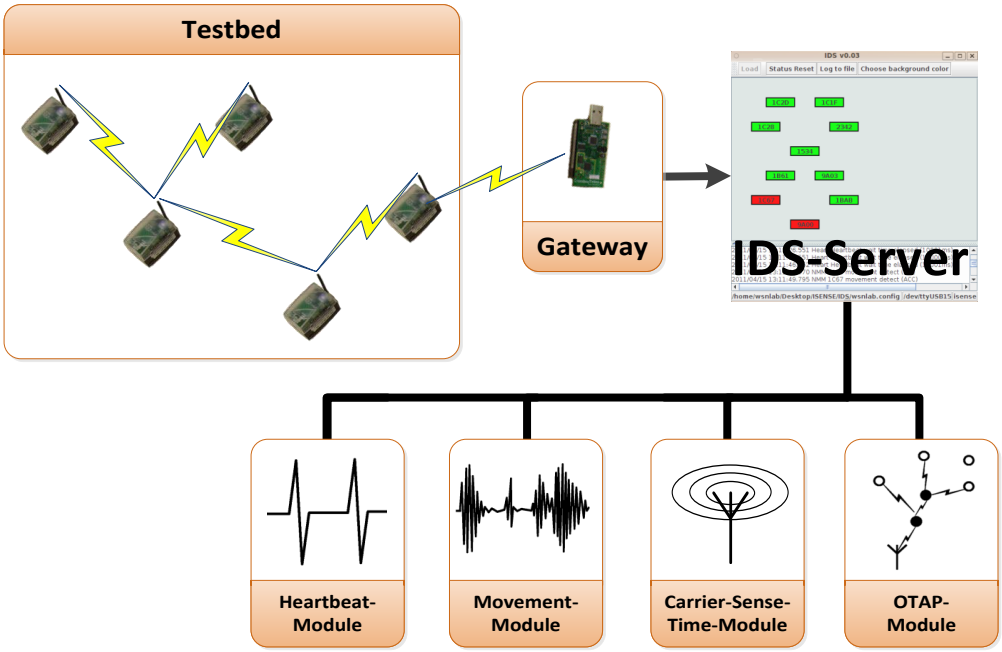


Abbildung 6.1: Schematische Darstellung der IDS-Architektur.

6.1.1 IDS-Architektur

Das IDS ist in lokale, sich auf den Sensorknoten befindende IDS-Komponenten und in einen zentralen Server unterteilt, der auf der Basisstation des WSN-Betreibers ausgeführt wird. Die Kommunikation innerhalb dieses Systems verläuft ausschließlich von den IDS-Komponenten in Richtung des Servers in Form von spezieller IDS-Nachrichten. Zu diesem Zweck ist der IDS-Server über einen Gateway-Knoten an das Sensornetz angebunden und verfügt über eine Reihe diverser Module zur Verarbeitung unterschiedlicher IDS-Nachrichten. Abbildung 6.1 skizziert diese Architektur.

Die lokalen IDS-Komponenten sind in den entsprechenden Betriebssystemen der Sensorknoten implementiert und in die bestehende Applikation des Sensorknotens integriert. Es kann zwischen zwei grundlegenden Typen von Komponenten unterschieden werden. Auf der einen Seite gibt es Komponenten, die in IDS-Nachrichten periodisch Statusinformationen an den Server übermitteln, wie beispielsweise der Heartbeat-Sensor, der als Gegenmaßnahme gegen den Remove Node Angriff (siehe Abschnitt 5.6.2) realisiert wurde. Auf der anderen Seite existieren Komponenten, deren IDS-Nachrichten ereignisbasiert generiert und übertragen werden. Ist das auslösende Ereignis zur Generierung der Nachrichten die Detektion eines potentiellen Angriffs innerhalb des Sensornetzes, werden diese Komponenten als Detektoren bezeichnet. Ein Beispiel für einen solchen Detektor ist die Signaturverifikations-Routine der sicheren OTAP-Anwendung. Eine fehlerhafte Signatur deutet auf einen Angriff hin und wird durch eine IDS-Alarmnachricht dem OTAP-Modul des IDS-Servers gemeldet. Um zwischen verschiedenen Komponente bzw. deren Status- oder Alarmnach-

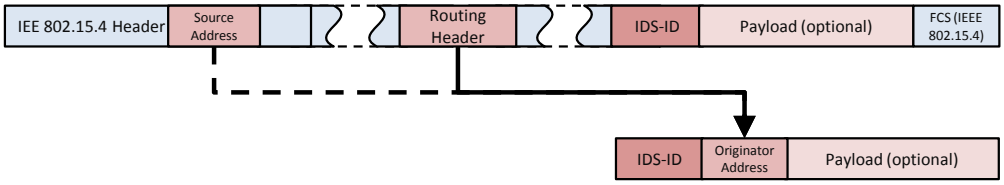


Abbildung 6.2: Transformation von IDS-Nachrichten im Gateway zwischen Testbed und IDS-Server. Das IDS-Nachrichtenformat der UART-Schnittstelle (unten) wird gegenüber dem IDS-Nachrichtenformat im IEEE 802.15.4 Netzwerk (oben) um die Adresse des Absenders erweitert.

richten differenzieren zu können, ist jeder Komponente eine eindeutige IDS-ID zugeteilt. Diese Kennung gibt darüber hinaus Auskunft über die Länge der jeweiligen IDS-Nachricht, die neben der ID optionale Nutzdaten enthalten kann, vgl. Abbildung 6.2.

Die Übertragung der IDS-Nachrichten einzelner Komponenten an den zentralen Server wird durch Gateway-Knoten ermöglicht. Dazu ist für jedes der drei realisierten Teilnetze ein Gateway vorgesehen. Diese Gateway-Knoten fungieren als Datensenke der IDS-Komponenten in dem entsprechenden Teilnetze und sind bei der Weiterleitung der auf der Funk-Schnittstelle empfangenen Nachrichten über die serielle UART-Schnittstelle an den Server für eine adäquate Transformation der Nachrichten verantwortlich. Diese Transformation ist in Abbildung 6.2 dargestellt. Neben dem Entfernen aller nicht weiter benötigten Protokollheader, wie beispielsweise dem IEEE 802.15.4 Header und dem Header eines Routing-Protokolls, sieht die Transformation vor, dass das IDS-Nachrichtenformat auf der UART-Schnittstelle um ein Adressfeld erweitert wird. In dieses Adressfeld wird die Adresse des Absenders der Nachricht eingetragen, die der Gateway-Knoten entweder aus dem Header des Routing-Protokolls bezieht oder direkt aus dem Quelladressfeld des IEEE-Headers, falls bei der Übertragung kein Routing-Protokoll eingesetzt wurde. Anhand dieser zusätzlichen Information kann der Server daraufhin die Zuordnung zum Ursprung der Nachricht vornehmen.

Die Aufgabe des IDS-Servers ist die Auswertung eingehender Nachrichten. Der Server ist als Java Applikation implementiert und somit mittels Java Runtime Environment (JRE) in allen gängigen Betriebssystemen lauffähig. Seine Funktionalität wird im nachfolgenden Abschnitt erläutert.

6.1.2 Funktionalität des IDS-Servers

Der IDS-Server ist das Ziel aller IDS-Nachrichten, die über den Gateway-Knoten empfangen werden. Damit unterschiedliche Nachrichten geeignet verarbeitet werden können, sieht das Design des Server die Integration separater Module vor. An diese Module werden eingehende Nachrichten von der Empfangskom-

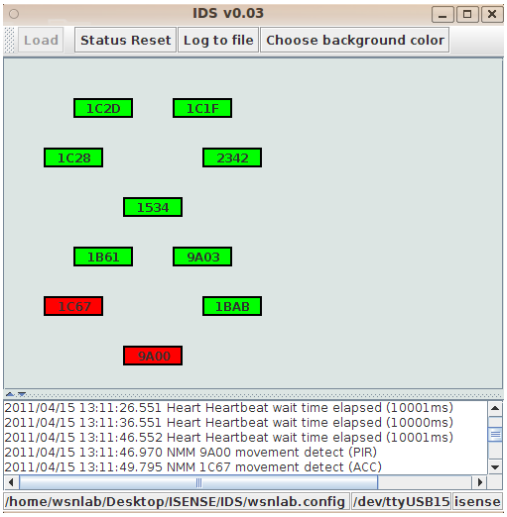


Abbildung 6.3: Screenshot der IDS-GUI.

ponente des Servers zur Verarbeitung intern übergeben, die daraufhin auf den Eingang oder auf den Inhalt der Nutzlast der Nachricht reagieren können.

Die Klasse der Module lässt sich in Module unterteilen, welche einer bestimmten IDS-ID fest zugeordnet sind und in Module, die im *promiscuous mode* alle Nachrichten erhalten. Diese Unterteilung ist durch den unterschiedlichen Informationsbedarf der Module motiviert. Einige Module sind nicht auf einen fixen IDS-Nachrichtentypen beschränkt. So interpretiert beispielsweise das Heartbeat-Modul jede eingehende Nachricht als Lebenszeichen des Nachrichtenabsenders. Andere Module sind dagegen ausschließlich zur Verarbeitung eines fixen Nachrichtentypens vorgesehen, wie beispielsweise das Movement-Module zur Alarmierung bei Bewegung von Sensorknoten.

Zusätzlich zur Verteilung eingehender Nachrichten stellt der IDS-Server eine graphische Benutzeroberfläche (Graphical User Interface (GUI)) bereit. Ein Screenshot dieser Oberfläche ist in der Abbildung 6.3 gegeben. Die Benutzeroberfläche besteht aus zwei Elementen. Das erste Element bildet die Visualisierung des zu überwachenden Sensornetz in dem Hauptfenster der Benutzeroberfläche. Dabei werden alle Sensorknoten des Netzes mit einem Identifikationsnamen auf einer festen Positionen dargestellt. Beide Informationen – die Identifikation und die Knotenposition – sind frei wählbar und über eine Konfigurationsdatei im Vorfeld zu definieren, die in die Benutzeroberfläche geladen wird. So kann sich bei der Identifikation eines Sensorknotens beispielsweise für eine Beschreibung seiner Funktion oder wie in der Abbildung 6.3 für seine 2 Byte IEEE-Adresse entschieden werden. Des Weiteren unterstützt die Konfigurationsdatei eine Sensorknoten-spezifische Zuordnung von Modulen anhand der Identifikation. Das bedeutet, dass sich Module bei Bedarf für eine Teilmenge von Sensorknoten aktivieren lassen. Standardmäßig wird die Identifikation jedes Sensorknotens in der Visualisierung mit der Farbe Grün hinterlegt. Zur direkten visuellen Rückmeldung wechselt die Farbe zu Rot, sobald eines der für den Knoten zuständigen Module einen Alarm generiert. Nach Kenntnisnahme

Modul:	ID:	Nutzdaten:	Informationsgehalt:
Heartbeat	*	beliebig (insb. Sensordaten)	Lebenszeichen (*: jede beliebige (IDS-) Nachricht)
Movement	1	auslösende Sensorik (PIR/ACC)	Movement-Detektion
CST	2	Alarmstatus, CST-Sample, Schwellwert	Jamming-Detektion bzw. Änderung des Alarmzustands
OTAP	3	Image Nr.	fehlgeschlagene Signaturverifikation
	4	Counter	Initialisierung eines OTAP-Prozesses mit invalidem Counter
	5	–	(unautorisiertes) (Re-)Booten
	6	–	Timeout während OTAP-Prozess
	7	Counter	erneutes Initialisieren einer OTAP-Operation während laufendem Prozess

Tabelle 6.1: IDS-Nachrichten der in Abbildung 6.1 skizzierten IDS-Module und deren jeweiliger Informationsgehalt.

des Alarms kann der Farbwechsel durch den Endnutzer manuell zurückgesetzt werden.

Als zweites Element befindet sich ein Protokolltextfeld in der Benutzeroberfläche, in dem die Alarmnachricht der Module zusammen mit dem Zeitpunkt des Auftretens ausgegeben wird. Des Weiteren existiert neben der unmittelbaren Ausgabe der Status- und Alarmnachrichten im Protokolltextfeld die Option, die Ausgabe in einer Datei zu protokollieren, um eine spätere Analyse der aufgetretenen Ereignisse zu ermöglichen.

6.1.3 IDS-Module

Die bisher realisierten Module, die dem IDS-Server zur Verfügung stehen, sind in der Abbildung 6.1 skizziert. Sie bestehen aus Heartbeat-, Movement-, CST- sowie OTAP-Modul. Zur Übersicht ist das Format der IDS-Nachrichten, die von diesen Modulen verarbeitet werden, in der Tabelle 6.1 dargestellt. Zusätzlich sind jeweils exemplarische IDS-IDs dieser Nachrichten zusammen mit der optionalen Verwendung des Nutzdatenfelds und dem Informationsgehalt in der Tabelle aufgeführt. Die tatsächlich in der Umsetzung des Testbeds eingesetzten IDs weichen von den in der Tabelle dargestellten Werten ab und erlauben eine Zuordnung der Nachricht zu dem jeweiligen Teilnetzwerk. Die IDS-Module wurden in das bestehende Intrusion Detection System integriert. Die beiden erstgenannten Module (Heartbeat- und Movement-Modul) werden in diesem Abschnitt zusammen mit ihrer Umsetzung in der Laborumgebung vorgestellt. Bei den beiden letztgenannten Modulen handelt es sich um Module, denen je ein bestimmter Detektor zugeordnet ist. Sie dienen lediglich der Visualisierung der von den lokalen Detektoren ausgehenden Alarm-Nachrichten. Aus diesem Grund wird an dieser Stelle nicht näher auf die Module eingegangen. Stattdessen werden die zugehörigen Detektoren und die generierten IDS-Nachrichten in den Abschnitten 6.3 und 6.5 detailliert beschrieben.

Heartbeat-Modul

Das Heartbeat-Modul ist ein promiscuous mode Modul, das Zugriff auf sämtliche eingehenden Nachrichten der zugeordneten Sensorknoten besitzt. Das Modul dient der allgemeinen Erkennung von Verbindungsverlusten und Knotenausfällen, insbesondere jedoch der Detektion von Remove Node Angriffen, siehe Abschnitt 5.6.2.

Innerhalb des Moduls wird der Eingangszeitpunkt von Nachrichten der überwachten Sensorknoten kontrolliert. Es wird für jeden dieser Knoten beobachtet, ob in einem vordefinierten Zeitintervall, das durch einen globaler Timer realisiert ist, Nachrichten von diesem Knoten empfangen werden oder ausblieben. Erreichen den Server ausgehend von einem überwachten Knoten innerhalb des Zeitfensters keine Nachrichten, ist ein Verbindungsverlust, ein Knotenausfall oder ein Remove Node Angriff wahrscheinlich. Aus diesem Grund wird in einem solchen Fall umgehend eine Alarmnachricht von dem Modul an den Server zur Ausgabe in der Benutzeroberfläche zurückgegeben.

In der realisierten Testumgebung wurde eine exemplarische Sensorapplikation implementiert (vgl. Abschnitt 3.3.1), in der Sensorknoten mit einer Periode von 5 Sekunden gewonnene Sensordaten an eine Basisstation übertragen. Gemäß Abschnitt 5.6.2 dient die periodische Übertragung zugleich der für das Modul benötigten Statusinformation (Heartbeats). Aus diesem Grund wurde die Größe des Timers der Senderate der Anwendung angeglichen. Grundsätzlich ist die Wahl der Timer-Größe t maßgebend für die Verzögerung der Detektion eines potentiellen Remove Node Angriffs. Aufgrund des globalen Zeitfensters ergibt sich folgende Ungleichung für die Detektionsverzögerung Δ:

$$t < \Delta < 2 \cdot t.$$

Diese Ungleichung ist unabhängig davon, ob das promiscuous mode Modul neben der periodischen Übertragung weiteren Datenverkehr berücksichtigt. Für die in der Testumgebung umgesetzte Anwendung bedeutet obige Abschätzung, dass sich ein Remove Node Angriff frühstens nach 5 Sekunden, spätestens jedoch nach 10 Sekunden erkennen lässt. Ist eine frühzeitigere Detektion gewünscht, so ist es zusammen mit der Anpassung der Timer-Größe erforderlich, neben der Sensordatenübertragung zusätzliche periodische Heartbeats zu versenden. Dies impliziert eine höhere Auslastung des Sensornetzes und insbesondere einen gesteigerten Energiebedarf der Sensorknoten. Aus diesem Grund ist für eine konkrete Anwendung zwischen der Detektionsverzögerung und der Energieeffizienz abzuwägen.

Movement-Modul

Das Movement-Modul ist ein ereignisbasiertes Modul, dem eine feste IDS-ID zugeteilt ist, und dient der Erkennung von Dislocate Node Angriffen in statischen Sensornetzen (siehe Abschnitt 5.6). Die Funktion des Moduls ist die Verarbeitung eingehender Movement-Nachrichten. Dabei wird untersucht,

welche Sensorik den Alarm welcher lokalen Komponente ausgelöst hat. Das Ergebnis wird anschließend zurück an den Server übergeben. Zur Verdeutlichung sind in Abbildung 6.3 zwei Alarmnachrichten des Movement-Moduls aufgeführt: Der Knoten mit dem Identifikationsnamen `9A00` meldet eine über die Passive InfraRed (PIR)-Sensorik detektierte Bewegung, während der Knoten `1C67` einen durch den Beschleunigungssensor (engl. Accelerometer (ACC)) ausgelösten Alarm mitteilt.

Der umgesetzte Laboraufbau umfasst drei Teilnetzen (vgl. Abschnitt 3.3). Die im TinyOS- und Contiki-Teilnetz eingesetzte TelosB-Plattform verfügt weder über einen Beschleunigungssensor noch über PIR-Sensorik. Beide Sensortypen sind jedoch zur Realisierung der zum Movement-Modul gehörigen lokalen IDS-Komponente erforderlich. Aus diesem Grund konnte die Movement-Komponente lediglich das iSense-Netzwerk integriert werden, dessen Sensorknoten mit der erforderlichen Sensorik ausgestattet sind. Die Movement-Komponente registriert Bewegungen über den Beschleunigungs- und den PIR-Sensor und versendet bei Detektion umgehend einen Alarm an den IDS-Server. Die Sensitivität der Sensorik ist dabei konfigurierbar.

6.1.4 Fazit

In diesem Abschnitt wurde das Intrusion Detection System vorgestellt, dass die Kernkomponente der Sicherheitsarchitektur und die Schnittstelle für den Endnutzer darstellt. Es wurde auf eine Reihe möglicher IDS-Module eingegangen. Unter diesen Modulen befinden sich Komponenten, die zur Verarbeitung von Alarmnachrichten von lokalen Detektoren dienen, die im Kapitel 5 als Gegenmaßnahme zu den analysierten Bedrohungen konzipiert sind. Die Umsetzung dieser Detektoren wird im weiteren Verlauf des Kapitels beschrieben. Durch das modulare Design können über das Projekt hinaus weitere Module in das IDS integriert werden.

6.2 Sinkhole

Der Sinkhole-Angriff ist ein Angriff auf Routing-Ebene mit hohem Einfluss auf die Kommunikation, da der Angreifer so viel Datenverkehr wie möglich auf sich zieht und so Pakete nach Belieben mitlauschen, manipulieren oder verwerfen kann (siehe Abschnitt 5.5.1). Zur detaillierten Betrachtung wurde das CTP [61, 66] als Routing-Protokoll gewählt. CTP ist weit verbreitet im Bereich der WSNs und im Gegensatz zu RPL [157] weniger generisch und auf intra-WSN-Kommunikation zugeschnitten.

Im Folgenden wird zunächst auf die Berechnung der Link-Qualitäten in CTP eingegangen (Abschnitt 6.2.1). Eine Kenntnis darüber ist notwendig, um die Implementierung des Sinkhole-Angriffes (Abschnitt 6.2.2) besser verstehen zu können. Abschließend wird ein Fazit gezogen und ein Ausblick auf weiterführende Arbeiten gegeben (Abschnitt 6.2.3).

6.2.1 Berechnung der Link-Qualitäten

CTP ist ein *adressierungs-freies* Routing-Protokoll, welches *Anycast*-Kommunikation zu einem der Root-Knoten bereitstellt [61]. „Adressierungs-frei“ bedeutet, dass ein Sensorknoten einen Root-Knoten nicht explizit als Empfänger adressiert, sondern diesen implizit durch den nächsten Hop auswählt. Dabei ist auch weniger wichtig, einen bestimmten Root-Knoten zu erreichen, sondern mehr, eine kürzeste Route zu einem beliebigen Root-Knoten zu haben („Anycast“). Motiviert durch das Convergecast-Kommunikationsmuster (vgl. Abschnitt 2.2.1) ist bei CTP nur die *Route zum Root-Knoten* von Interesse. Sensorknoten bilden demnach anhand der Pfadmetrik einen logischen Routing-Wald (bestehend aus mehreren Routing-Bäumen) mit Root-Knoten als Wurzeln. Die Pfadmetrik setzt sich dabei aus der Link-Metrik der einzelnen Links additiv zusammen.

Als Link-Metrik kommt ETX zum Einsatz. ETX ist die erwartete Anzahl an Paketen, die gebraucht werden für eine erfolgreiche Übertragung inklusive eines Link-Layer-ACKs vom Empfänger und wurde ursprünglich von De Couto et al. für MANETs vorgeschlagen [41]. Die am weitesten verbreitete Formel zur Berechnung ist $ETX = 1/(LQ_{in} \cdot LQ_{out})$, wobei $LQ_{in} \in [0, 1]$ die eingehende, und $LQ_{out} \in [0, 1]$ die ausgehende uni-direktionale Link-Qualität bezeichnet. Die Berechnung des ETX-Werts in CTP ist durch das Link Estimation Exchange Protocol (LEEP) spezifiziert [65]. Das TinyOS Enhancement Proposal (TEP) Dokument ist allerdings nicht auf dem neuesten Stand der tatsächlichen Implementierung. Folgende Beschreibung basiert daher hauptsächlich auf Analyse des Codes in `tos/lib/net/le` und `tos/lib/net/ctp` von TinyOS 2.1.1.

Der aktuelle ETX-Wert für einen (bi-direktionalen) Link wird mit einem Exponentially Weighted Moving Average (EWMA) Filter berechnet, d.h. der alte Mittelwert (ETX_{i-1}^{avg}) geht ebenso wie der neu berechnete Wert (ETX_i) anteilig in die Berechnung des neuen Mittelwerts (ETX_i^{avg}) ein:

$$ETX_i^{avg} = \alpha \cdot ETX_{i-1}^{avg} + (1 - \alpha) \cdot ETX_i$$

wobei $\alpha \in \mathbb{R}$, $0 < \alpha < 1$, der Alterungsfaktor ist, mit dem die Gewichtung reguliert werden kann (Standardwert $\alpha := 0,9$). Für $ETX_i \in \mathbb{R}_{>=1}$ stehen 16 Bit zur Verfügung und der Wert wird (wie alle anderen auch) im Code als Festkommazahl dargestellt, da keine Fließkommaoperationen möglich bzw. zu teuer sind. Zur Vereinfachung wird hier allerdings von dieser Code-spezifischen Darstellung abstrahiert. Initial gilt $ETX_0^{avg} := 1$, d.h. der Link wird zu Beginn als optimal angenommen. Dies führt allerdings zu unnötigen temporären Änderungen in der Topologie, falls ein Knoten neu hinzukommt, dessen reale Link-Qualität schlecht ist. ETX_i wird aus zwei Quellen gewonnen: Zum einen aus der Broadcast-Driven Link Estimation (LE) und zum anderen aus der Data-Driven LE.

Die *Broadcast-Driven* LE geht zurück auf die in [65] definierten Nachbarschaftsinformationen. Jeder Knoten A berechnet für alle Nachbarn B, die in die Nachbarschaftstabelle passen (max. 10 Einträge aufgrund der Ressourcenknappheit),

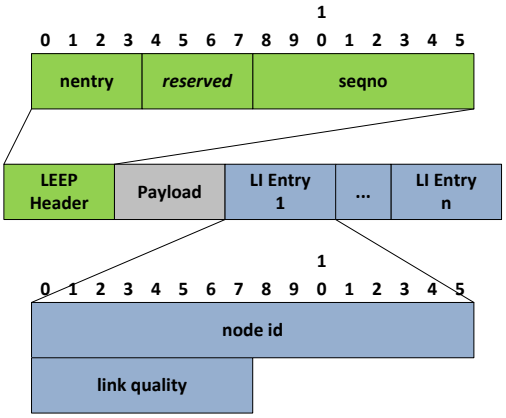


Abbildung 6.4: Paketformat eines LEEP-Frames (mittig) [65]. Im Header (oben) steht die Anzahl der Link Information Einträge (`nentry`) sowie die Sequenznummer (`seqno`) zur Bestimmung von verlorengegangenen Frames. Der Footer besteht aus Link Information Einträgen. Ein Link Information Eintrag (unten) enthält die Link-Layer-Adresse des entsprechenden Nachbarknotens (`node id`) und die eingehene Link-Qualität (`link quality`) bzgl. dieses Nachbarn.

die *eingehende* Link-Qualität $LQ(B, A)$, also die Qualität des Links von B nach A. Diese wird analog zu ETX mit einem EWMA Filter berechnet:

$$LQ(B, A)_i^{avg} = \alpha \cdot LQ(B, A)_{i-1}^{avg} + (1 - \alpha) \cdot LQ(B, A)_i$$

wobei α wie oben und $LQ(B, A)_i \in \mathbb{R}$, $0 \leqslant LQ(B, A)_i \leqslant 1$, als 8 Bit Indikator für die Link-Qualität (größerer Wert bedeutet bessere Qualität). Initial gilt $LQ(B, A)_0^{avg} := 0$, d.h. die eingehende Link-Qualität wird zu Beginn als pessimal angenommen. Dies steht zwar im Gegensatz zur Initialisierung des ETX-Werts, erscheint aber sinnvoller, da sich so der eingehende Link eines neu hinzugekommenen Knotens erst als hochwertig erweisen muss, bevor er ausgewählt wird. $LQ(B, A)_i$ wird wie folgt berechnet:

$$LQ(B, A)_i = Bcst(B)_{rcvd} / (Bcst(B)_{rcvd} + Bcst(B)_{fail})$$

d.h. die Ankunftsrate der periodischen (LEEP-)Broadcast-Frames von B bestimmt die eingehende Link-Qualität. Die Anzahl der nicht empfangenen Broadcasts ($Bcst(B)_{fail}$) kann mithilfe der fortlaufenden Sequenznummer (`seqno`), im LEEP Header (vgl. Abbildung 6.4) bestimmt werden, da hier bei Paketverlust eine Lücke entsteht. $LQ(B, A)_i$ wird jedesmal nach `BLQ_PKT_WINDOW` (erfolgreich) empfangenen Broadcasts neu berechnet, d.h. $Bcst(B)_{rcvd}$ = `BLQ_PKT_WINDOW` (Standardwert 3). Um nun die Broadcast-Driven LE für A zu vervollständigen, wird noch die Link-Qualität für den Link von A nach B, also $LQ(A, B)_i$ benötigt. Diese erhält A als Teil der LEEP-Broadcast-Frames von B, denn in diesen sind alle eingehenden Link-Qualitäten aus Sicht von B (Link Information Einträge, vgl. Abbildung 6.4) enthalten, also insbesondere

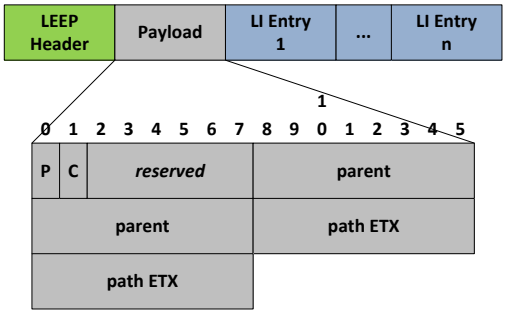


Abbildung 6.5: Paketformat eines CTP-Routing-Frames (unten) [61]. Das Pull Bit (`P`) fordert Routing-Frames von benachbarten Knoten an und löst daher bei Empfang einen Reset des Routing-Frame Trickle Timer aus. Weiterhin wird ein Advertisement des aktuellen Elternknotens (`parent`) und die dazugehörige Pfad-ETX (`path ETX`) mit jeweils 16 Bit verbreitet.

auch $LQ(A,B)_i$. Die beiden uni-direktionalen Link-Qualitäten werden bei jeder neuen Berechnung von $LQ(B,A)_i$ zur bi-direktionalen Link-Qualität ETX_i aggregiert:

$$ETX_i = 1/(LQ(B,A)_i \cdot LQ(A,B)_i)$$

Dieser Wert fließt dann wiederum in die Berechnung von ETX_i^{avg} ein.

Die *Data-Driven* LE soll die Broadcast-Driven LE komplementieren, indem anhand der tatsächlichen (CTP-)Datenpakete ein Wert für ETX_i berechnet wird. Damit ist eine tatsächliche Messung des ETX-Werts gegeben, denn die Broadcast-Driven LE kann streng genommen nur als Näherung betrachtet werden. Allerdings ist die Broadcast-Driven LE Voraussetzung dafür, denn Datenpakete können erst versendet werden, wenn erste ETX-Werte und somit der nächste Hop bestimmt sind. Eine weitere Voraussetzung sind Link-Layer-ACKs, ansonsten kann nicht überprüft werden, ob ein Datenpaket erfolgreich versendet wurde oder nicht. Mit der Data-Driven LE wird ein neuer ETX_i-Wert wie folgt berechnet:

$$ETX_i = Data_{tx}/Data_{ack}$$

wobei $Data_{tx}$ die Anzahl versendeter Datenpakete und $Data_{ack}$ die Anzahl ACKs bezeichnet. Die Berechnung erfolgt jedesmal nach `DLQ_PKT_WINDOW` versendeten Datenpaketen, d.h. $Data_{tx}$ = `DLQ_PKT_WINDOW` (Standardwert 5). Auch hier fließt der Wert wiederum in die Berechnung von ETX_i^{avg} ein. So werden Broadcast-Driven und Data-Driven LE in einem Wert aggregiert, obwohl die Berechnung unabhängig voneinander erfolgt.

`BLQ_PKT_WINDOW` und `DLQ_PKT_WINDOW` steuern also die (logische) Frequenz einer neuen Berechnung von ETX_i. Zeitlich gesehen hängt dies von der *Sendeperiode* der Broadcasts bzw. Datenpakete ab. Einfluss auf die Datensendeperiode hat nur die Anwendung selbst und kann demnach nicht direkt vom Routing-Layer bestimmt werden. Die Sendeperiode der LEEP-Frames hängt von der Sendeperiode der CTP Routing-Frames ab, da letztere die Payload eines LEEP-Frames

darstellen (vgl. Abbildung 6.5). Im Wesentlichen steuert ein Trickle (exponentiell wachsender randomisierter) Timer [99,101] das Versenden von CTP Routing-Frames. Ein Reset dieses Timers wird hervorgerufen durch den Empfang eines Routing-Frames, in dem das `P` Bit gesetzt ist (vgl. Abbildung 6.5) und falls beim Routen-Update die Differenz in der Pfad-ETX zwischen aktuellem Elternknoten und bestem Nachbarn (bzgl. Pfad-ETX) einen gewissen Schwellwert überschreitet.

Routen-Updates werden u.a. periodisch (Parameter `BEACON_INTERVAL`) und vor jedem Senden eines Routing-Frames durchgeführt. Dabei wird anhand verschiedener Kriterien die beste Route unter allen bekannten ausgewählt. Das wichtigste Kriterium ist, dass die beste Route bzw. der beste Nachbar mindestens um einen gewissen Schwellwert (Parameter `PARENT_SWITCH_THRESHOLD`) besser sein muss als die aktuelle Route bzw. der aktuelle Elternknoten. Dies soll verhindern, dass durch geringe Schwankungen in der Link-Qualität ständig zwischen verschiedenen Nachbarn gewechselt wird. Als Metrik wird hier ausschließlich die Pfad-ETX zu einem Root-Knoten zugrunde gelegt. Diese setzt sich additiv aus der vom Nachbarn per Advertisement angekündigten Pfad-ETX (vgl. Abbildung 6.5) und der zu diesem Nachbarn gehörenden, wie oben beschrieben berechneten Link-ETX (ETX_i^{avg}) zusammen.

6.2.2 Implementierung

Als Realisierungsmethode für den Sinkhole-Angriff wurde das klassische Fälschen von Link-Qualitäten gewählt (weitere Methoden wurden in Abschnitt 5.5.1 behandelt). Der größte Unterschied zu den aus MANETs bekannten Sinkhole-Angriffen ist, dass aufgrund der verschiedenen Kommunikationsmuster (Peer-to-Peer im Vergleich zu Convergecast) andere Kriterien für eine attraktive Route gelten: Zählen bei Peer-to-Peer noch eine hohe (1-Hop-)Link-Qualität und eine große Nachbarschaft, so zählt bei Convergecast hauptsächlich die Qualität des Pfads zur Senke. Letztere lässt sich i.A. nicht fälschen, da der legitime Elternknoten seine eigene Pfad-Qualität kennt und eine Topologieinkonsistenz annimmt, falls einer seiner Kindknoten plötzlich ein CTP Datenpaket mit einer besseren Pfad-Qualität an bzw. über ihn versenden will. Der Elternknoten reagiert darauf mit einem Reset des Trickle Timers für ein sofortiges Routen-Update sowie einem randomisierten Backoff für die Weiterleitung des Pakets. D.h. ein mote-class Angreifer (von dem hier ausgegangen wird) ist abhängig von der Pfad-Qualität seines Elternknotens und kann lediglich die (1-Hop-)Link-Qualität zu diesem fälschen.

Die generelle Idee bei der Implementierung ist, nur Code für den Angreifer zu ändern, da sonst keine allgemeine Anwendbarkeit des Angriffs gegeben wäre. Da die eingehenden Link-Qualitäten zu den einzigen Werten gehören, die nur vom Angreifer berechnet und von den Nachbarn nicht ohne weiteres überprüft werden können, kommen diese auch für das Fälschen in Frage. Belässt man es dabei, sind die Chancen jedoch sehr gering, einen Sinkhole-Angriff zum Erfolg zu bringen. Wie im vorangegangenen Abschnitt ausführlich beschrieben,

werden die durch LEEP-Frames verbreiteten Link-Qualitäten immer erst nach `BLQ_PKT_WINDOW` empfangenen Broadcasts zu einem neuen Link-ETX-Wert verrechnet. Dabei macht es keinen Unterschied, ob alle dieser empfangenen Broadcasts oder nur der letzte eine optimale Link-Qualität enthält, denn es wird stets der zuletzt empfangene Wert verwendet. Dazu kommt, dass die neu berechnete Link-ETX nur mit geringem Gewicht in den EWMA Filter für ETX_i^{avg} eingeht (Standardwert $(1-\alpha) = 0,1$). Weiter kommt erschwerend hinzu, dass der Trickle Timer für die Broadcasts exponentiell anwächst und die Data-Driven LE auf Dauer die gefälschte Link-Qualität wieder zu nichte machen könnte. Dies alles führt zu einem trägen Prozess, der nicht mal einen Angriffserfolg verspricht.

Um die Erfolgschancen zu erhöhen, werden daher auch die Link-ETX-Werte gefälscht. Diese werden zwar nicht propagiert wie die eingehenden Link-Qualitäten, werden aber als Summand in die Berechnung der Pfad-ETX miteinbezogen. Letztere wird als Teil des CTP-Routing-Frames propagiert (vgl. Abbildung 6.5). Ein Kindknoten des Angreifers kann i.A. nicht überprüfen, ob die von letzterem propagierte Pfad-ETX korrekt ist, denn die darin enthaltene gefälschte Link-ETX bezieht sich auf den Link zwischen Angreifer und dessen Elternknoten. Auf diese Weise kann also sowohl der bi-direktionale Link zwischen Angreifer und Elternknoten als auch der von Kindknoten des Angreifers eingehende Link als optimal gefälscht werden.

Ein weiterer wichtiger Aspekt ist das lokale Rechnen mit den gefälschten Werten: Es reicht nicht aus, die Link-Qualitäten und -ETX-Werte kurz vor dem Versenden der Broadcasts in den entsprechenden Feldern (`link quality` bzw. `path ETX`, vgl. Abbildung 6.4 bzw. 6.5) mit den gefälschten zu überschreiben. Der Angreifer muss sich sozusagen selbst etwas vormachen, indem er bewusst mit den gefälschten Werten in seinen internen Datenstrukturen arbeitet und damit z.B. ein Routen-Update oder einen Inkonsistenz-Check durchführt. Ansonsten kann es dazu kommen, dass ein Kindknoten eine Pfad-ETX propagiert, die nach realen Werten besser ist als die des Angreifers und so zu einer Inkonsistenz in der Topologie führt. Intern mit gefälschten Werten zu rechnen hat den Nachteil, dass der Angreifer seinen Elternknoten nur noch anhand der propagierten Pfad-ETX auswählt (die Link-ETX ist dann für alle Nachbarn optimal). Dies ist nicht im Sinne einer robusten Übertragung, aber umso mehr im Sinne des Angriffs, denn so wird automatisch der Nachbar mit der niedrigsten Pfad-ETX gewählt, welche – wie weiter oben erklärt – nicht problemlos gefälscht werden kann.

Da für die Implementierung des Angriffs Basis-Code von TinyOS geändert werden musste, entscheidet ein Präprozessor-Flag (`SINKHOLE`) über das Einschalten des relevanten Codes. So ist sichergestellt, dass für legitime Knoten derselbe Basis-Code verwendet werden kann. Der Angriff lässt sich auf Anwendungsebene über eine Erweiterung des TinyOS `CtpInfo` Interface starten und beenden:

```
1 interface CtpInfo {
2   #ifdef SINKHOLE
3   /* enable sinkhole attack */
4   command error_t enableSinkhole();
5
6   /* disable sinkhole attack */
7   command error_t disableSinkhole();
```

```
 8
 9    /* get function for sinkhole flag */
10    command bool isSinkholeActive();
11    #endif
12     ...
13  }
```

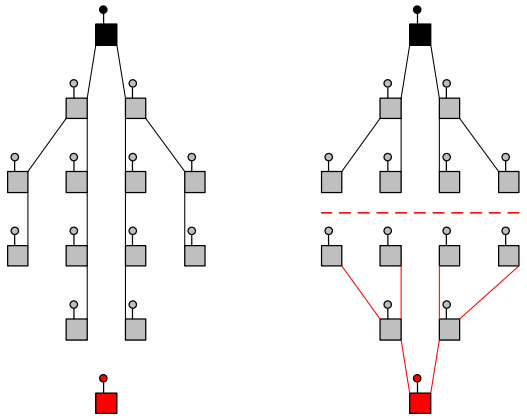
Um nach Beenden des Angriffs die Nachbarschaftstabelle nicht von neuem aufbauen zu müssen, wird vor dem Start des Angriffs eine Kopie der eingehenden Link-Qualitäten sowie der Link-ETX-Werte gemacht. Diese wird auch während des Angriffs immer weiter mit den realen Werten aktualisiert. Wird der Angriff über die obige Application Programming Interface (API) gestoppt, werden die im Hintergrund aktualisierten Werte wieder übernommen.

Damit die gefälschten Werte auch sofortigen Einfluss auf die Routenwahl des Angreifers selbst haben, wird der Routing-Frame Trickle Timer neu gestartet. Dies löst neben einem Routen-Update auch das Versenden von LEEP-Frames aus, welche die aktualisierte Route sowie die gefälschten Werte an die Nachbarn propagieren. Das einmalige Setzen des Pull Bits (vgl. Abbildung 6.5) stellt zudem sicher, dass die Nachbarknoten ebenfalls ihren Trickle Timer neu starten, was den gleichen Effekt wie beim Angreifer hat und die evtl. nun über den Angreifer laufende Route weiter im Netzwerk propagiert. Eine Erfolgsgarantie für den Angriff gibt es jedoch nicht, denn der Angreifer muss eine Route anbieten mit einer Pfad-ETX, welche um mindestens `PARENT_SWITCH_THRESHOLD` besser ist als die der aktuellen Route der Nachbarn (vgl. hierzu auch den vorigen Abschnitt).

Praktische Tests der Implementierung haben gezeigt, dass eine Evaluation im realen Netz, wie sie ursprünglich geplant war, nicht durchführbar ist. Dies liegt einerseits an der Robustheit bzw. Trägheit des Protokolls und andererseits an den durch die Umgebung bedingten Path-Loss-Effekten des Ausbreitungssignals und den daraus resultierenden unvorhersehbaren Link-Qualitäten. So war es nicht möglich, Sensorknoten so zu platzieren, dass die gewünschte logische Topologie (insbesondere inkl. Angriff) daraus entsteht oder gar reproduzierbar ist. Dies macht eine sinnvolle Evaluation mit Replikationen und dem Vergleich verschiedener Angriffsvarianten oder gar Erkennungsmethoden unmöglich.

6.2.3 Fazit und Ausblick

Eine ausführliche Analyse von CTP hat aufgezeigt, dass die klassische Realisierung des Sinkhole-Angriffs, nämlich das Fälschen der eingehenden Link-Qualität, keine großen Erfolgschancen verspricht. Daher wurde die Implementierung um das Fälschen der Link-ETX zwischen Angreifer und dessen Elternknoten erweitert. Aufgrund von Robustheit bzw. Trägheit des Protokolls sowie durch die Umgebung bedingter Path-Loss-Effekte konnten Angriffe nicht zuverlässig reproduziert werden. Von einer Implementierung der ursprünglich vorgesehenen Challenge-Response-Methode als Gegenmaßnahme für gefälschte Link-Qualitäten wurde abgesehen, da dies nur einer partiellen Erkennung des



(a) vorher (b) nachher

Abbildung 6.6: Root Node Spoofing: (a) zeigt den Topologie-Baum vor, (b) nach dem Angriff.

realisierten Angriffs entsprechen würde und darüber hinaus der Angriff nicht zuverlässig Auswirkung zeigte.

Ein Spezialfall des Sinkhole-Angriffs, welcher größere Erfolgschancen verspricht und zudem keine Änderungen im Basis-Code benötigt, ist das *Root Node Spoofing*. Hierbei gibt sich der Angreifer als Root-Knoten aus, welche per Definition die beste Pfad-ETX besitzen, und nutzt die Anycast-Kommunikation von CTP zu seinem Vorteil. Wie zu Beginn von Abschnitt 6.2.1 erklärt, wählen die Knoten in CTP automatisch einen beliebigen Root-Knoten, zu dem sie die kürzeste Route haben. Nachteilig beim Root Node Spoofing ist allerdings, dass es sich nicht um einen (allgemeinen) Sinkhole-, sondern lediglich um einen Blackhole-Angriff handelt, bei dem die empfangenen Pakete schlicht verworfen werden. Bei dieser Methode sollte sich der Angreifer so tief wie möglich im Routing-Baum befinden, damit die Differenz in der Pfad-ETX zu legitimen Root-Knoten maximiert wird. Eine exemplarische Topologie ist in Abbildung 6.6 dargestellt: Vor dem Angriff (Abbildung 6.6(a)) senden alle (grauen) Knoten an den (schwarzen) Root-Knoten. Hat der Angreifer (rot) sich als Root-Knoten ausgegeben, so wählen die Knoten, für welche die Differenz in der Pfad-ETX zum Angreifer groß genug ist, nun diesen als Senke (Abbildung 6.6(b)). So wird der ursprüngliche Baum in zwei geteilt (angedeutet durch die rot gestrichelte Linie). Als Gegenmaßnahme ist der Einsatz von TSNs als Root-Knoten denkbar, so dass der Angreifer von anderen Knoten gar nicht erst als Senke in Betracht gezogen wird.

Beim allgemeinen Sinkhole ist zur Erhöhung der Erfolgschancen weiterhin denkbar, den ETX-Schwellwert zum Elternknotenwechsel (`PARENT_SWITCH_THRESHOLD`) herabzusetzen. Dies würde allerdings Änderungen im Code für legitime Knoten voraussetzen und ist daher i.A. auch als unzulässige Alternative einzustufen. Davon abgesehen könnte eine solche Änderung zudem negative Auswirkungen auf die Netzwerk-Performance haben,

da auch bei weniger signifikanten Schwankungen in der Link-Qualität häufiger die Routen gewechselt werden.

Um einen höheren Einfluss der gefälschten eingehenden Link-Qualitäten zu erzielen, könnte der Angriff so erweitert werden, dass nicht nur der Trickle Timer zu Angriffsbeginn zurückgesetzt, sondern die Frequenz der Broadcasts generell erhöht wird. Damit ginge der gefälschte Wert häufiger in den EWMA Filter für die Link-ETX ein, was wiederum den Angriff beschleunigen könnte. Zudem könnte dies die Data-Driven LE kompensieren, welche die tatsächliche ETX misst und nicht vom Angreifer beeinflusst werden kann. Eine Evaluation dieser oder weiterer möglicher Erweiterungen des Angriffs gehen jedoch über den Rahmen dieses Projekts hinaus und werden daher als mögliche zukünftige Arbeiten angesehen. Im Rahmen der in Abschnitt 6.6.4 durchgeführten simulativen Evaluation wird jedoch noch gezeigt, dass Routing-Angriffe prinzipiell, gerade bei größeren Netzen, durchaus einen signifikanten Einfluss haben können.

6.3 Jamming

Der Jamming-Angriff ist ein Angriff auf der physikalischen Ebene des OSI-Modells, dessen Ziel es ist, die drahtlose Kommunikation durch eine physikalische Störübertragung zu behindern (vgl. Abschnitt 5.3.2). Im Rahmen des Konzeptes der Sicherheitsarchitektur wurde zur Erkennung von Jamming-Angriffen eine lokale Anomalie-Erkennung vorgesehen (siehe Abbildung 5.2 bzw. Kapitel 5). Vermeidende Gegenmaßnahmen wie beispielsweise das Frequenzsprungverfahren werden im Rahmen dieses Berichts dagegen nicht untersucht, stellen aber interessante weiterführende Arbeiten dar. In diesem Abschnitt wird die Implementierung der lokalen Anomalie-Erkennung beschrieben und die Ergebnisse der Evaluation dieser Erkennung präsentiert.

6.3.1 Implementierung des Carrier-Sense-Time-Sensors

Es gibt verschiedene Indikatoren, die zur Detektion einer Anomalie im Kontext von Jamming-Angriffen verwendet werden können. Dies sind z.B., wie bereits in Abschnitt 5.3.2 beschrieben, die Signalstärke (RSSI) empfangener Signale, die erwartete Paketankunftsrate (engl. PDR) und die Carrier-Sense-Time (CST), also die Zeit, die bei einem Sendeversuch für den Zugriff auf das gemeinsame Medium benötigt wird. Aufgrund der hohen Varianz der RSSI-Werte und dem im Vergleich zu mobilen ad-hoc Netzwerken (MANETs) geringen Datenverkehr in WSNs eignen sich die beiden erstgenannten Indikatoren (RSSI und PDR) nur bedingt für den Einsatz in Sensornetzwerken. Weil darüber hinaus Erfahrungen im Bereich von MANETs gezeigt haben, dass die CST ein geeigneter Indikator ist (vgl. [9]), wurde sich im Rahmen des Projekts auf diesen Indikator als Metrik für den Jamming-Detektor beschränkt. In [9] ist zudem ein Verfahren zur lokalen Jamming-Detektion vorgestellt und simulativ evaluiert worden, das sich adaptiv an die Last bzw. Überlast im Netzwerk anpasst. Die-

ses schwellwert-basierte Verfahren wurde zur Umsetzung und Integration in das IDS der Sicherheitsarchitektur ausgewählt.

In dem Verfahren werden CST-Werte (Samples s) eines Netzwerkteilnehmers kontinuierlich mit einer fixen Abtastrate ermittelt. Anhand dieser Samples wird ein adaptiver Schwellwert zur Anpassung an die Netzwerklast und zur Unterscheidung zwischen gewöhnlicher Last und Anomalien fortwährend aktualisiert. Die Berechnung des aktuellen Schwellwerts t_n ergibt sich mit Hilfe folgender Alterungsfunktion aus dem bisherigen Schwellwert t_o, der zu Beginn mit dem Wert 0 initialisiert wird:

$$t_n = t_o \cdot (1 - \alpha) + (s \cdot \alpha). \tag{6.1}$$

Das Verhältnis der Gewichtung von aktuellem und bisherigem Schwellwert ist in dieser Alterungsfunktion durch den Parameter α festzulegen. Ein Alarm, der die potentielle Detektion eines Jamming-Angriffs signalisiert, wird ausgelöst, sobald ein aktuell gemessener Sample um mehr als die mit dem Parameter w gewichteten Standardabweichung σ von dem bis zu diesem Zeitpunkt aktuellen Schwellwert t_o abweicht, d.h. es gilt:

$$alarm := \begin{cases} 1, & s > t_o + \sigma \cdot w \\ 0, & \text{else.} \end{cases} \tag{6.2}$$

Der Toleranzbereich basierend auf der Standardabweichung, die sich wie folgt aus den einzelnen CST-Samples s_i, $i \in \{1, ..., n\}$, berechnet, dient dabei dem Ausgleich von natürlichen Schwankungen in der CST:

$$\sigma = \sqrt{\frac{1}{n-1}\left[\sum_{i=1}^{n}(s_i^2)) - \frac{1}{n}(\sum_{i=1}^{n}(s_i))^2\right]}. \tag{6.3}$$

Die IDS-Komponente des lokalen Jamming-Detektors reagiert auf Veränderungen des Alarmzustands des CST-Sensors. Bei jeder Änderung des Alarmzustands wird eine IDS-Nachricht generiert und ein Versuch unternommen, diese an den zentralen IDS-Server zu versenden. Diese Nachricht enthält im Nutzdatenfeld neben dem aktuellen Alarmzustand den gemessenen CST-Sample und den momentanen Schwellwert, inklusive der gewichteten Standardabweichung $\sigma \cdot w$ (siehe Tabelle 6.1).

Der Jamming-Detektor wurde sowohl in Contiki als auch in TinyOS implementiert. Zur effizienten und speicherplatz-sparsamen Implementierung der Berechnung der Standardabweichung in obiger Alarmfunktion waren dabei für die in den entsprechenden Teilnetzen der Laborumgebung eingesetzten, ressourcenbeschränkten Sensorknoten-Plattformen (TelosB und MicaZ) einige algorithmische Optimierungen erforderlich, da die Microcontroller dieser Plattformen keine Gleitkommaeinheit (Floating-Point Unit (FPU)) besitzen. In beiden Betriebssystemen wird standardmäßig auf MAC-Ebene ein CSMA-Protokoll verwendet, dessen Carrier-Sense-Mechanismus zur Bestimmung der CST genutzt werden kann. Aufgrund der im Allgemeinen in WSNs – und insbesondere auch in der bestehenden Laboranwendung (periodische Sensordatenübertragung alle

5 s) – geringen Last, wird der Carrier-Sense-Mechanismus verhältnismäßig selten beansprucht. Daher eignet sich dieser nicht alleine zur Ermittlung der CST und es wurde zusätzlich das optionale Radio Duty Cycling (RDC) (vgl. Abschnitt 2.1.2) für die erforderliche CST-Messung berücksichtigt. Die Berücksichtigung des RDCs bietet zwei Vorteile. Zum einen wird zur CST-Berechnung keine zusätzliche Energie benötigt, zum anderen ermöglicht sie Sensorknoten, die wenig Sendeversuche unternehmen wie beispielsweise dem Gateway-Knoten, ohne weitere Anpassung die Berechnung der CST. Unabhängig davon, ob ein CST-Sample durch CSMA oder RDC gemessen wird, ist dieser definiert als das Zeitintervall zwischen Beginn der Messung und dem Erkennen eines freien Mediums. Übersteigt die Dauer dieses Intervalls das Intervall der Abtastrate, wird diese Dauer auf den folgenden Sample addiert. Dies beutet, dass in einer Phase, in der das Medium kontinuierlich belegt ist, die ermittelten CST-Werte streng monoton ansteigen. Das Addieren sichert also das Anwachsen der Werte in einer derartigen Phase. Im Folgenden wird auf die OS-spezifischen Besonderheiten der Implementierungen eingegangen.

Contiki-CST-Sensor

Im Contiki-OS wurde der CST-Sensor gemäß des oben beschriebenen Verfahrens in das Standard-MAC-Protokoll ContikiMAC integriert, das für das RDC zuständig ist. Eine Änderung des mit der ContikiMAC verknüpften CSMA-Protokolls ist in diesem Betriebssystem nicht notwendig, da dieses CSMA-Protokoll für das Carrier-Sensing die verfügbare Funktionalität von ContikiMAC nutzt. Für jede auf der MAC-Ebene aufbauende Schicht ist die getätigte Änderung des ContikiMAC-Codes transparent. Alle Protokolle über der MAC-Ebene sind durch die Integration des CST-Sensors unbeeinflusst. Dies erlaubt eine anpassungsfreie Verwendung bestehender Protokolle.

TinyOS-CST-Sensor

Standardmäßig ist das RDC in TinyOS (LPL) deaktiviert. Wie bereits erwähnt, ist jedoch beim Einsatz in Sensornetzen das Einbeziehen des RDCs in die Ermittlung der CST-Samples sinnvoll, um gerade in Netzwerken mit geringer Aktivität einen zuverlässigen Detektor zu realisieren. Aus diesem Grund ist die Aktivierung von LPL für den CST-Sensor in TinyOS standardmäßig aktiviert, lässt sich bei Bedarf jedoch optional deaktivieren. Die LPL-Komponente ist in die beiden Module DefaultLPL und PowerCycle unterteilt. Das letztgenannte Modul ist für den eigentlichen Carrier-Sense-Mechanismus verantwortlich. Die Funktionalität dieses Mechanismus wurde zur Integration des CST-Sensors in das TinyOS-System genutzt, der analog zur Contiki-Implementierung umgesetzt wurde.

6.3.2 Evaluation

Zur Leistungsbewertung und der Evaluation des Verhaltens der realisierten Jamming-Detektors in Abhängigkeit der Parametrisierung des Alterungsparameters α und der Gewichtung der Standardabweichung w, wurden verschiedene Messreihen unter Einfluss eines Jamming-Angriffs in der Laborumgebung durchgeführt. Bei der Messung wurde ein TelosB-Knoten mit dem in TinyOS implementierten CST-Sensor verwendet, der zur Auswertung der CST-Samples, des berechneten Alarmschwellwerts und des Alarms über die UART-Schnittstelle mit einem Computer verbunden ist. Der Sensorknoten wurde zwei Jamming-Phasen eines ebenfalls über den Computer gesteuerten, konstanten Jammers (vgl. Abschnitt 5.3.2) ausgesetzt. Der Einfluss dieser Jamming-Phasen auf die Detektion, d.h. sowohl auf den Alarmschwellwert als auch auf die Dauer der Detektion, wurde in verschiedenen Messreihen für unterschiedliche Parameterbelegungen untersucht. Analog zu [9] wurde dazu $w \in \{1; 2; 3; 4; 5\}$ und $\alpha \in \{0,5; 0,1; 0,05; 0,01; 0,005; 0,001\}$ gewählt. Die Dauer jeder Messreihe, die in folgende Phasen unterteilt ist, beträgt 105 s. Jede Messreihe beginnt mit einer Ruhephase von 20 s, in der es keine Aktivität im Netzwerk gibt. Danach folgt die erste Jamming-Phase von 30 s, an die eine erneute Ruhephase von 30 s anschließt. Als nächstes wird die zweite Jamming-Phase mit der Länge von 15 s ausgeführt. Abschließend gibt es eine letzte Ruhephase von 10 s. Zudem geht jeder Messreihe eine Ruhephase von 100 s voraus, die als Einschwingphase dient. Da innerhalb dieser Phase keine Auffälligkeiten beobachtet wurden, wird auf die Einschwingphase im weiteren Verlauf der Evaluation nicht erneut eingegangen. Der konstante Jamming-Angriff der beiden Jamming-Phasen der Evaluation wurde mit Hilfe des im Rahmen des Laboraufbaus realisierten IEEE 802.15.4 Jammers (siehe Abschnitt 3.3.5) ausgeführt.

Die Abbildung 6.7 zeigt das Ergebnis einer exemplarischen Messreihe in Abhängigkeit der Zeit mit der Parametrisierung $\alpha = 0,001$ und $w = 5$. Das Störsignal des Jammers behindert während der beiden Jamming-Phasen (20–50 s und 80–95 s) die drahtlose Kommunikation. Das Medium wird in dieser Zeit als belegt eingestuft, was zu einer linear ansteigenden CST (linke y-Achse) führt. Nach Deaktivieren des Jammers, ist der Zugriff aufs Medium wieder möglich. Die CST beträgt in einem freien Medium nur wenige ms. Neben der CST ist in der Abbildung die zeitliche Entwicklung des in Abhängigkeit von α aus der CST resultierenden Alarmschwellwerts visualisiert. Gemäß der ersten Bedingung $s > t_o + \sigma \cdot w$ in Formel 6.2 wurde zur Veranschaulichung zusätzlich die gewichtete Standardabweichung auf den Schwellwert (rechte y-Achse) addiert. Zu erkennen ist, dass der durch die erste Jamming-Phase bedingte Anstieg des Schwellwerts eine Verzögerung der Detektion der zweiten Jamming-Phase verursacht, die in der Abbildung durch den rötlichen Alarm-Bereich dargestellt ist.

Neben der Parametrisierung des Alarmschwellwerts durch den Alterungsparameter α nimmt auch die Gewichtung w der Standardabweichung Einfluss auf die Verzögerung der Jamming-Detektion, da ein aktuell ermittelter CST-Sample

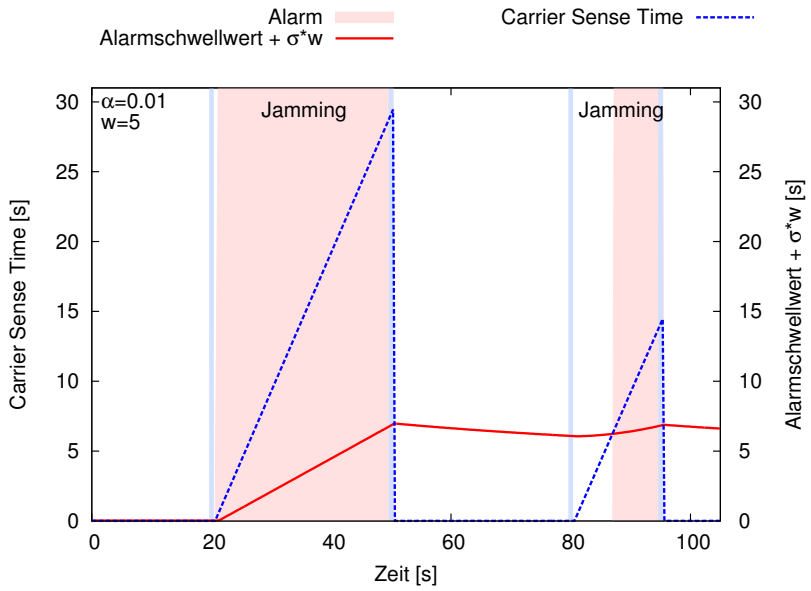
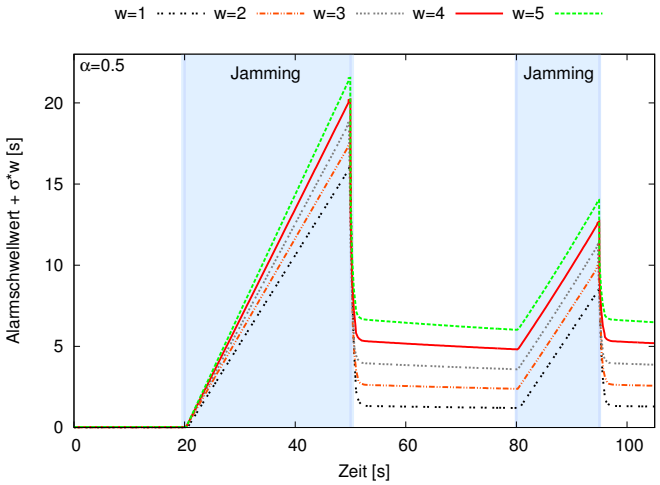


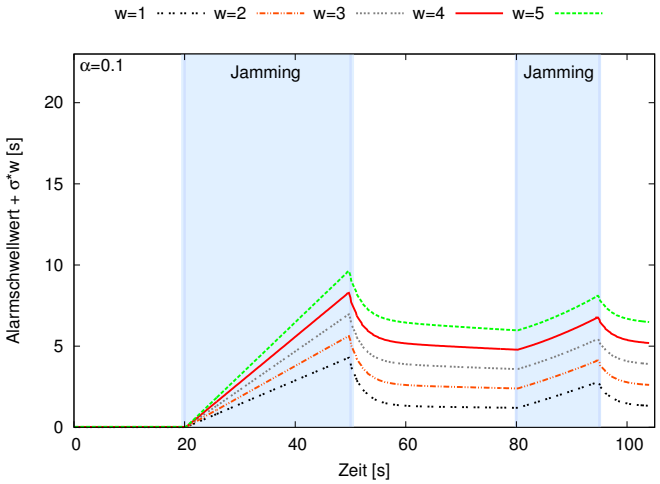
Abbildung 6.7: Die aus der ermittelten CST und dem berechneten Schwellwert resultierende Jamming-Detektion während zwei Jamming-Phasen in einer exemplarischen Messreihe ($\alpha = 0,001$ und $w = 5$).

mehr als die gewichtete Standardabweichung von dem Schwellwert abweichen muss, damit ein Alarm ausgelöst wird. Abbildung 6.8 visualisiert den Einfluss der Parametrisierung auf den Alarmschwellwert, auf den hier analog zur vorherigen Abbildung zusätzlich die gewichtete Standardabweichung addiert wurde. Die Abbildung bestätigt, dass eine stärkere Gewichtung der Standardabweichung w zu einem höheren derartigen Schwellwert führt und damit zu einer höheren Toleranz gegenüber Fluktuationen in der CST. Andererseits ist die Erkennung bei großem w nicht so sensible. Es kann somit verstärkt zu nicht erkannten Angriffen kommen. Des Weiteren verdeutlicht der Vergleich der Abbildungen 6.8(a) und 6.8(b), die Auswirkung des Alterungsparameters α. Je geringer dieser Parameter ist, desto stärker ist die Gewichtung der Historie des Schwellwerts und desto träger reagiert die Adaption des Schwellwerts auf Veränderungen in der CST. Die Auswirkungen der in Abbildung 6.8 dargestellten Parameterkombinationen auf die Verzögerung der Detektion der Jamming-Angriffe in den durchgeführten Messreihen ist in der Abbildung 6.9 durch die Dauer der Detektion beider Angriffe veranschaulicht.

Unabhängig von der getroffenen Parameterwahl wird der erste Jamming-Angriff fast in voller Länge (30 s) detektiert. Die Detektionverzögerung liegt dabei stets unter 0,8 s. Die Ursache für diese Beobachtung liegt in der Einschwing- und der ersten Ruhephase, in der keine Netzwerkaktivitäten existieren. Lediglich geringe Interferenzen erzeugen in diesen Phasen vernachlässigbare Fluktuationen in der CST. Daher beträgt der Alarmschwellwert zu Beginn der ersten Jamming-Phase nahezu den Wert 0. Das gleiche gilt für die Standardabweichung zu diesem Zeitpunkt. Sobald der Angriff ausgeführt wird und die CST



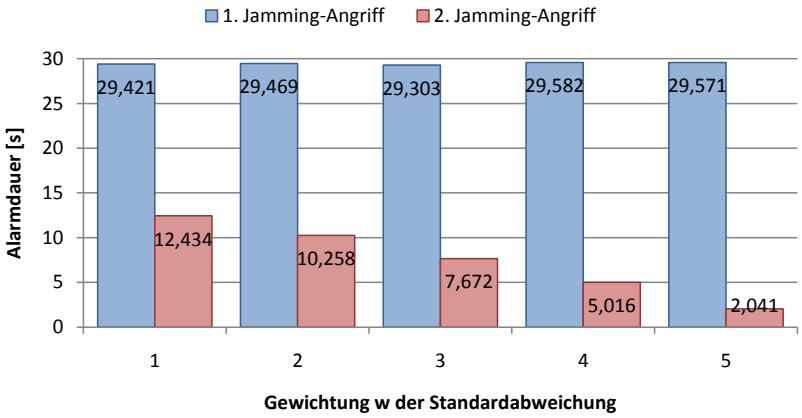
(a) Alterungsparameter $\alpha = 0,5$



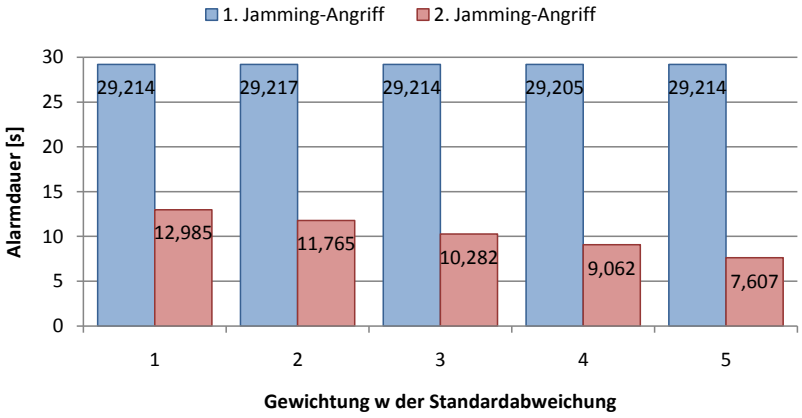
(b) Alterungsparameter $\alpha = 0,1$

Abbildung 6.8: Einfluss der Parametrisierung des Jamming-Detektors auf den Alarmschwellwert.

linear ansteigt, wird der Alarm ausgelöst. Die Betrachtung der Detektionsdauer in der zweiten Jamming-Phase (15 s) bestätigt dagegen die bisherigen Ergebnisse. Eine stärkere Gewichtung der Standardabweichung führt auf der einen Seite zu einer höheren Fluktuationstoleranz, auf der anderen Seite daher jedoch zu einer späteren Detektion. Dagegen ergibt eine stärkere Gewichtung der Schwellwert-Historie (geringeres α) eine höhere Trägheit des Schwellwerts, die sich in dem betrachteten Szenario durch die lange initiale Ruhephase positiv auf die Detektionsdauer bzw. -geschwindigkeit auswirkt.



(a) Alterungsparameter $\alpha = 0,5$



(b) Alterungsparameter $\alpha = 0,1$

Abbildung 6.9: Einfluss der Parametrisierung des Jamming-Detektors auf die Alarmdauer.

6.3.3 Fazit und Ausblick

In diesem Abschnitt wurde zunächst die realisierte Implementierung des auf der CST-Messung basierenden Jamming-Detektors beschrieben und anschließend Ergebnisse der Evaluation vorgestellt. Die Evaluation des Jamming-Detektors hat die grundlegende Funktion des adaptiven Detektors gezeigt. Dabei wurden die Einflüsse der Parametrisierung auf das Detektionspotential des Detektors untersucht, welche eine anwendungsbedingte Konfigurationsmöglichkeit bietet. Es ist zu erwarten, dass aus einer stärkeren Gewichtung der Standardabweichung und der Schwellwert-Historie erhöhte False Negatives auftreten, während bei der umgekehrten Gewichtung mit häufigen False Positives zu rechnen ist. Insbesondere in Sensornetzen mit hohen Lastspitzen wäre diesbezüglich eine weiterführende Untersuchung von Interesse.

6.4 Kryptographische Verfahren

In diesem Abschnitt soll auf die kryptographischen Verfahren eingegangen werden, die im Rahmen der Sicherheitsarchitektur zum Einsatz kommen. Ausschlaggebend für die Verwendung eines Algorithmus ist vor allem die Ressourcenbeschränktheit der Sensorknoten. Im Folgenden wird zunächst auf die symmetrische Verschlüsselung eingegangen. Anschließend wird die asymmetrische Verschlüsselung beschrieben. Jeder dieser Abschnitte ist gegliedert in theoretische Betrachtungen, die Implementierung und anschließende Evaluation.

6.4.1 Symmetrische Kryptographie

Im Allgemeinen zeichnet sich symmetrische Kryptographie dadurch aus, dass sowohl zum Entschlüsseln als auch zum Verschlüsseln der gleiche Schlüssel verwendet wird. Es wird unterschieden zwischen Strom- und Blockchiffre. Bei einer Stromchiffre wird der Klartext byteweise verschlüsselt, ein Blockchiffre verarbeitet den Klartext in Blöcken vordefinierter Größe. Als Trivialbeispiel für eine Stromchiffre kann das One-Time-Pad genannt werden. Hier wird der Klartext mit einem mindestens genauso langen, zufälligen Schlüssel kombiniert um die Chiffre zu erhalten. Mit Hilfe verschiedener Modi lässt sich auch eine Blockchiffre in eine Stromchiffre überführen. Beispielsweise lassen sich einmalige Blöcke, auch Nonces[1] genannt, verschlüsseln und dann ähnlich wie beim One-Time-Pad mit dem Klartext per XOR verknüpfen. Die Nonces werden aus Parametern generiert, die deren Einzigartigkeit für einen Block gewährleisten sollen, wie z.B. einem Zähler. Dieser Modus wird auch Counter Modus (CTR) genannt. Die so entstehenden verschlüsselten Zähler haben zwar weiterhin eine feste Blockgröße, jedoch kann für die Kombination mit dem Klartext auch nur ein Teil eines Blocks verwendet werden. Zum Entschlüsseln werden die gleichen Nonces generiert und verschlüsselt, um anschließend den Klartext aus dem Chiffre extrahieren zu können.

Da TelosB und MicaZ Knoten mit dem CC2420 Funkchip [159] ausgerüstet sind, der eine Hardwareimplementierung von AES bereitstellt, ist eine symmetrische Verschlüsselung auf Basis von AES prädestiniert für Sensorknoten. Der CC24020 Funkchip erlaubt eine Ver- und Entschlüsselung von Paketen on-the-fly, d.h. die Verschlüsselung wird direkt auf dem Sende- bzw. Empfangspuffer ausgeführt, was zusätzliche Datentransfers überflüssig macht. Die zum Einsatz kommenden Verschlüsselungsmethoden sind der CTR Modus sowie der CBC-MAC Modus zur Erstellung von Signaturen. Eine Kombination beider Modi ist ebenfalls möglich und nennt sich Counter mit CBC-MAC (CCM) Modus. Die Signaturen umfassen neben der eigentlich Nutzlast ebenfalls den Paket-Header mit Quell- und Zieladresse sowie der Sequenznummer. Dadurch kann sichergestellt werden, dass das Paket nicht unbemerkt verändert werden kann, um es bspw. in einem Replay-Angriff zu verwenden.

[1] nonce: engl. Abkürzung für *number used once*

Eine derart erstellte Signatur ist aber nicht mit der Signatur durch ein asymmetrisches Verfahren vergleichbar, da mit einer Signatur durch eine symmetrische Verschlüsselung lediglich die Integrität von Daten sichergestellt werden kann. Grundsätzlich lässt sich mit symmetrischer Kryptographie zwar auch das Merkmal der Authentizität realisieren, jedoch ist der Aufwand sehr hoch und das System kann schnell unübersichtlich und damit fehleranfällig werden. Konkret muss für jedes Paar von Kommunikationspartnern und jede Kommunikationsrichtung ein eindeutiger symmetrischer Schlüssel existieren. Im schlimmsten Fall, d.h. jeder kommuniziert mit jedem anderen, werden für n Teilnehmer insgesamt $\frac{n \cdot (n-1)}{2}$ Schlüssel benötigt, je n davon muss jeder Teilnehmer kennen und verwalten. Ein weiteres Problem ist die Schlüsselverteilung. Diese muss gesichert (im Sinne von Vertraulichkeit, Authentizität und Integrität) sein, da allein das Mitlesen des Schlüssels einem Angreifer erlaubt, Nachrichten für den entsprechenden Kommunikationsweg erfolgreich zu fälschen oder zu entschlüsseln. Eine sinnvolle Alternative bietet hier die asymmetrische Kryptographie, deren Einsatzmöglichkeiten in Abschnitt 6.4.2 beschrieben werden.

Trotz der vorhandenen Hardwareimplementierung von AES durch den CC2420 Funkchip stellt sich die Frage, ob andere Methoden nicht doch günstiger sein können. Hansen et al. [69] haben jedoch gezeigt, dass die Implementierung mittels Hardware deutlich effizienter ist als eine Implementierung des gleichen AES Verschlüsselungsalgorithmus in Software. Die besondere Effizienz der Hardwareimplementierung wird vor allem deutlich, wenn man die 128-bit Variante mit einer Softwareimplementierung von Skipjack [116] (mit einer Blockgröße von 64-bit) vergleicht. Während Skipjack im Allgemeinen performanter ist als eine Softwareimplementierung des AES Algorithmus, reicht sie trotz der geringeren Blockgröße nicht an die Hardwareimplementierung heran. Zu dem gleichen Ergebnis kommen auch Jinwala et al. [86]. Des Weiteren zeigen Law et al. [98], dass auch andere Verschlüsselungsmethoden nicht hinreichend effizient mit einer Softwareimplementierung realisiert werden können.

Neben der Verschlüsselung des Netzverkehrs lassen sich mit dem CC2420 Funkchip ebenfalls Daten im CTR Modus verschlüsseln, ohne dass diese sofort als Paket versendet werden. Dies erlaubt bspw. das Verschlüsseln von Daten, die anschließend im Flashspeicher abgelegt werden. Der Energie- und Zeitaufwand für diesen Vorgang wird jedoch maßgeblich durch das Schreiben in den Speicher des CC2420 Funkchips und dem anschließenden Auslesen beeinflusst, so dass obige Vorteile deutlich schwächer ausfallen. Der zusätzliche Bedarf an Random Access Memory (RAM) sowie Read-Only Memory (ROM) ist hingegen vernachlässigbar, da der Funkchip ohnehin verwendet werden muss (schlichtweg um Daten senden und empfangen zu können). Es stellt sich also die Frage, ob zur Verschlüsselung von Daten, die im Flashspeicher abgelegt werden sollen, eine zusätzliche Softwareimplementierung nicht sinnvoll ist, um Energie zu sparen. In der Praxis stellt man jedoch schnell fest, dass dies an der sehr knapp bemessenen Größe von RAM und ROM scheitert.

Implementierung

Im Rahmen der Implementierung wurde die Betriebssysteme TinyOS [163] und Contiki OS [47] sowie die Software Wireshark [178] modifiziert.

Für TinyOS und Contiki OS war eine Implementierung bzw. Anpassung der AES Verschlüsselung, genauer der Schnittstelle zum CC2420 Funkchip, die diese Funktionalität bereitstellt, notwendig. Während TinyOS einen Großteil der Schnittstelle bereits zur Verfügung stellte, mussten dennoch diverse Änderungen vorgenommen werden, um einerseits sicherheitsrelevante Mängel zu beheben, andererseits um eine nahtlose Integration der Verschlüsselung in andere Protokolle zu ermöglichen. Contiki OS dagegen stellte in Version 2.5RC1 lediglich den Zugriff auf die alleinige Verschlüsselungsfunktionalität bereit, ohne jedoch eine direkte und effiziente Verschlüsselung des Netzverkehrs zu ermöglichen. Sowohl für TinyOS als auch für Contiki OS mussten entsprechend tiefgreifende Änderungen in Kernbestandteilen des Betriebssystems vorgenommen werden. Ein Umstieg auf eine neuere Version erfordert demnach eine Portierung dieser Änderungen. In den folgenden beiden Abschnitten soll genauer auf die Details der einzelnen Probleme eingegangen werden.

Wireshark musste ebenfalls angepasst werden, damit Pakete korrekt entschlüsselt werden können. Die Änderungen beschränkten sich großteils auf die für die Entschlüsselung notwendigen Nonces, also Parameter die dem AES Blockchiffre übergeben werden. Wireshark zieht, dem IEEE 802.15.4 Standard folgend, einen 64-bit Extended Unique Identifier (EUI) des sendenden Knotens zur Erzeugung dieser Nonces heran. TinyOS verwendet diese Art von Adressen jedoch nicht, und setzt den entsprechenden Teil der Nonce auf 0. Durch eine Modifikation von Wireshark war es möglich, von TinyOS gesendete Pakete korrekt entschlüsseln zu können.

Änderungen in TinyOS

Die Implementierung der Schnittstelle zum CC2420 Funkchip war in TinyOS weitestgehend vollständig. Die vorgenommenen Änderungen betreffen zum einen sicherheitsrelevante Aspekte sowie Designentscheidungen, die eine Verwendung der Sicherheitsarchitektur unnötig erschweren.

Verschlüsselung des TinyOS Header: TinyOS verzichtet auf eine Verschlüsselung der betriebssystemeigenen Paketheader. Während dies grundsätzlich keine funktionale Einschränkung darstellt, werden hierdurch unnötigerweise Informationen über den Pakettyp offengelegt. Das kann zu einer erleichterten Kompromittierung durch einen Angreifer führen, da die Kenntnis über die Art des Pakets entscheidende Hinweise darauf geben kann, welche Pakete besonders wichtig sind. Die Spezifikation durch den IEEE 802.15.4 Standard ist diesbezüglich jedoch nicht eindeutig. So lassen sich beide Vorgehensweisen in die dort be-

schriebene Empfehlung interpretieren. TinyOS wurde modifiziert, so dass die komplette Nutzlast des IEEE 802.15.4 Frames verschlüsselt wird. Gemäß dem OSI-Schichtenmodell gehört hierzu der TinyOS-Header sowie die Header der Protokolle höherer Schichten.

Signieren des IEEE 802.15.4 Header: Analog zum vorherigen Fall wird in TinyOS der IEEE 802.15.4 Header nicht in die Signatur einbezogen, falls der CBC-MAC Modus verwendet wird. Der CBC-MAC Modus gibt an, dass das Paket nicht verschlüsselt, sondern lediglich dessen Integrität durch eine Signatur sichergestellt werden soll. Wird jedoch der IEEE 802.15.4 Header nicht mitsigniert, kann die korrekt signierte Nutzlast jederzeit von einem beliebigen Knoten als neues Paket versendet werden. Eine Erkennung aufgrund der Sequenznummer ist nicht möglich, da diese korrekt gesetzt werden kann, ohne das Paket erneut signieren zu müssen. Der IEEE 802.15.4 Standard ist dahingehend eindeutig, dass beginnend mit dem IEEE 802.15.4 Header signiert werden soll. TinyOS wurde modifiziert, so dass diese Vorgabe umgesetzt wird.

Verschlüsselung allgemein zugänglich machen: Das Design von TinyOS sieht vor, dass der Entwickler entscheidet, ob ein Knoten generell verschlüsselt kommunizieren soll. Dafür stehen für die Entwicklung zwei Module zur Verfügung, *AMSender* und *SecAMSender*, von denen letzteres die Verschlüsselung unterstützt. Wird sich bspw. bei einer Protokollimplementierung gegen Verschlüsselung entschieden, zieht eine spätere Änderung eine Anpassung der Implementierung in Bezug auf die Schlüsselwahl und Modus (verschlüsseln und/oder signieren) nach sich. Unter Umständen ist diese Anpassung mit größerem Aufwand verbunden und muss zudem für jedes Protokoll von neuem vorgenommen werden. Problematisch wird das vor allem, wenn mehrere Entwickler Programmcode beisteuern, z.B. wenn für das Routingprotokoll auf bestehende Lösungen zurückgegriffen wird. Die Lösung besteht darin, TinyOS in Kernbestandteilen zu ändern, sodass die Verschlüsselung erst auf unterster Ebene kurz vor dem Senden aktiviert wird. Bei der Übersetzung des Programms wird entschieden ob und mit welchen Parametern Verschlüsselung verwendet wird. Der Vorteil der Änderung auf unterster Ebene liegt darin, dass nun einzelne Protokollimplementierungen nicht angepasst werden müssen. Stattdessen wird die Verschlüsselung im Hintergrund und ohne weiteres Zutun der Protokollimplementierung angewandt.

Änderungen in Contiki OS

Die Änderungen in Contiki OS sind weitreichender. Wie eingangs kurz erwähnt stand hier allein die Funktionalität zur Verfügung, Daten im Speicher zu ver- und entschlüsseln. Zwar können diese Daten dann in einem Paket versendet werden, jedoch ist dieses Vorgehen äußerst ineffizient. Für das Ver- und Entschlüsseln von Paketen stellt der CC2420 Funkchip eine gesonderte Funktion bereit, die beide Vorgänge (verschlüsseln und senden bzw. empfangen und ent-

schlüsseln) in einer Operation durchführt. Die entsprechende Schnittstelle war in Contiki OS jedoch nicht implementiert. Dementsprechend wurden Kernbestandteile von Contiki OS modifiziert, so dass eine effiziente verschlüsselte Netzkommunikation möglich ist. Ebenso können mit Contiki OS signierte Nachrichten verschickt werden. Lediglich das Verifizieren von Paketsignaturen ist derzeit nicht möglich, ohne erhebliche Änderungen an der Struktur des Netzwerkstacks von Contiki OS vorzunehmen. Der Grund hierfür liegt in einer Designentscheidung seitens der Entwickler von Contiki OS, wodurch bei Paketempfang stets das gesamte Paket aus dem Empfangspuffer gelesen wird um es anschließend zu analysieren. Der Empfangspuffer wird dadurch direkt für weitere Pakete freigegeben. Nachteilig ist hierbei, dass Daten, deren Signatur durch den CC2420 Funkchip überprüft werden soll, nicht direkt ausgelesen und freigegeben werden dürfen. Stattdessen sollte zunächst der IEEE 802.15.4 Header kopiert werden, ohne den entsprechenden Pufferbereich freizugeben, um dann entscheiden zu können, welche Operationen (Verifikation einer Signatur, Entschlüsselung) für das Paket ausgeführt werden sollen. Entsprechende Modifikationen von Contiki OS sind zwar möglich, jedoch erschweren sie aufgrund ihres Umfangs einerseits die Eingliederung in zukünftige Versionen von Contiki OS, andererseits sind derartige Änderungen im Rahmen des Projektes nicht vorgesehen.

Evaluation

In diesem Abschnitt sollen die Auswirkungen der Verwendung von symmetrischer Verschlüsselung auf die Ressourcen eines Sensorknotens beschrieben werden. Hierfür werden zum einen die Auswirkungen auf den Speicherbedarf (sowohl RAM als auch ROM) sowie den Energiebedarf betrachtet. Nachdem die Messmethodik erläutert wurde, werden die Ergebnisse beschrieben.

Für jeden zu messenden Fall wurde ein Beispielprogramm in TinyOS auf TelosB Knoten mit jeweils verschiedenen Verschlüsselungsmodi als auch komplett ohne Verschlüsselung geprüft. Die Programme sind so aufgebaut, dass sie kontinuierlich entweder Datenpakete mit einer Nutzlast von 17 Byte versenden, oder diese Nutzlast auf den internen Flashspeicher schreiben. Dabei besteht die Nutzlast aus 16 festen Byte sowie einem Zähler von einem Byte. Im Gegensatz zum Schreiben in den Flashspeicher entsteht beim Versenden der Nutzlast ein zusätzlicher Platzbedarf durch die Protokoll-Header. Die Gesamtgröße des Pakets beträgt demnach 36 Byte bzw. 52 Byte, falls das Paket signiert wird. Die jeweiligen Programme für das Versenden bzw. Speichern unterscheiden sich lediglich bei der Umsetzung der Verschlüsselung. Hierbei sei erwähnt, dass der tatsächliche Unterschied zwischen zwei Varianten in konkreten Anwendungsfällen von den hier beschriebenen Unterschieden abweichen kann, da die Verwendung zusätzlicher Module eine Optimierung begünstigen kann. Folglich sind die aufgezeigten Messwerte als Richtwerte zu verstehen.

Der Bedarf an RAM und ROM wird beim Übersetzen des Programmcodes bestimmt und ausgegeben. Basierend auf diesen Informationen lassen sich Änderungen im Quelltext gut auf den Speicherbedarf abbilden. Der Energieverbrauch

der Knoten wurde mit einem handelsüblichen Multimeter (H&B Multavi 6) gemessen. Für Messwerte unter 1,5 mA kann die Messskala in Schritten von 0,025 mA, bis 6 mA in 0,1 mA abgelesen werden. Darüber hinaus beträgt die Genauigkeit lediglich 0,5 mA. Zwar ließen sich im letzteren Fall noch minimale und durchaus plausible Unterschiede erkennen, eine definitive Aussage lässt sich aufgrund der begrenzten Messgenauigkeit jedoch nicht treffen. Diese liegt bei dem verwendeten Gerät bei 1%. Als Stromquelle diente ein Schaltnetzteil, das eine Ausgangsspannung von 3 V liefert (WEC NTS 1500-312).

Eine Auflistung des verfügbaren RAM und ROM ist für verschiedene Sensorknoten in Tabelle 6.2 gegeben. Für iSense gibt es keine strikte Trennung zwischen RAM und ROM. Stattdessen wird beides in einem Speicher von 96 kB abglegt, wodurch allein die Summe aus RAM und ROM beschränkt wird.

Tabelle 6.3 zeigt die Ergebnisse einer Messung des Speicherbedarfs und des Stromverbrauchs für TelosB Knoten. In Klammern ist angegeben, welche Art der Verschlüsselung verwendet wurde. Bedingt durch den CC2420 Funkchip kann zum Speichern lediglich der CTR Modus verwendet werden. Sowohl bei der Betrachtung des Speicherbedarfs als auch beim Stromverbrauch fällt auf, dass die Unterschiede beim Senden von verschlüsselten und/oder signierten Paketen äußerst gering ausfallen. Auch die Entscheidung, ob überhaupt eine Art der Verschlüsselung verwendet wird, hat mit einem Mehrbedarf von rund 3,5 kB ROM und gut 100 Byte RAM nur geringe Auswirkungen. Die dem Stromverbrauch nachgestellten Plus- oder Minuszeichen deuten an, dass der Messzeiger leicht vor oder hinter den diskreten Teilstrich der Messskala zeigte.

Das Ablegen von Daten im Flashspeicher des Sensorknotens zeigt hingegen ein anderes Bild. Auch hier ist der Unterschied des Speicherbedarfs zwischen verschlüsseltem und unverschlüsseltem Speichern zwar sehr gering, jedoch wird durch die Verschlüsselung der Daten vor dem Speichern signifikant mehr Strom verbraucht. Während für bloßes Schreiben 2,4 mA benötigt werden, erhöht sich der Verbrauch mit vorangehender Verschlüsselung um mehr als den Faktor acht auf 20,5 mA. Der Grund hierfür liegt in der Verwendung des CC2420 Funkchips, da dieser die hardwareseitige Verschlüsselung zur Verfügung stellt. Das heisst, der Funkchip muss eingeschaltet sein, um die Verschlüsselung durchführen zu können, was zu dem hohen Verbrauch führt.

Abschließend werden die allgemeinen Auswirkungen einer Kombination von Modulen anhand eines konkreten Beispiels betrachtet. Bei den Modulen handelt es sich wie oben um ein Modul zum Speichern von Daten auf dem internen Flashspeicher und um ein Modul zum Senden von Paketen. Beide Module verschlüsseln die Daten wahlweise. Insgesamt ergeben sich dadurch folgende Modulkonfigurationen:

	TelosB	**MicaZ**	**iSense**
ROM	48 kB	128 kB	96 kB
RAM	10 kB	4 kB	

Tabelle 6.2: Verfügbarer Speicher auf unterschiedlichen Sensorknoten.

Messung	ROM-Bedarf	RAM-Bedarf	Stromverbrauch
Senden (ohne)	12.310 Byte	556 Byte	$18,5 \pm 0,3$ mA (-)
Senden (CTR)	15.566 Byte	662 Byte	$18,5 \pm 0,3$ mA (+)
Senden (CBC-MAC)	15.640 Byte	662 Byte	$18,5 \pm 0,3$ mA (+)
Senden (CCM)	15.644 Byte	662 Byte	$18,5 \pm 0,3$ mA (++)
Speichern (ohne)	8.756 Byte	369 Byte	$2,4 \pm 0,06$ mA
Speichern (CTR)	10.438 Byte	440 Byte	$20,5 \pm 0,3$ mA
idle	1.786 Byte	186 Byte	$0,02 \pm 0,003$ mA

Tabelle 6.3: Messergebnisse der verschiedenen Verschlüsselungsvarianten (in Klammern) für das Senden und Speichern von Daten. Die Fehlertoleranzen der Messungen des Stromverbrauchs basieren auf den Spezifikationen des Messgeräts.

- Daten im Flashspeicher ablegen
  - nichts speichern: es werden keine Daten im Flashspeicher abgelegt
  - Klartext speichern: die Daten werden unverschlüsselt im Flashspeicher abgelegt
  - verschlüsselt speichern: die Daten werden verschlüsselt im Flashspeicher abgelegt
- Daten senden
  - nichts übertragen: es werden keine Daten versendet
  - Klartext übertragen: die Daten werden unverschlüsselt versendet
  - verschlüsselt übertragen (CTR): die Daten werden verschlüsselt versendet
  - verschlüsselt übertragen (CBC-MAC): die Daten werden unverschlüsselt aber signiert versendet
  - verschlüsselt übertragen (CCM): die Daten werden verschlüsselt und signiert versendet

Tabelle 6.4 zeigt den Gesamtbedarf an ROM der verschiedenen Kombinationen. Anhand dieser Tabelle wird deutlich, dass sich die Kombination von Modulen unterschiedlich auf diesen Gesamtspeicherbedarf auswirkt. Schon in Tabelle 6.3 ist ein Unterschied des ROM-Bedarfs zwischen verschlüsseltem Speichern und Speichern von Klartext von 1.682 Byte erkennbar, wenn keine Daten übertragen werden. Betrachtet man hingegen einen Knoten, der auch Daten unverschlüsselt sendet, beläuft sich der Unterschied zwischen dem verschlüsseltem Speichern und Speichern von Klartext auf lediglich 784 Byte. Das heisst, der Umstand, dass der Knoten nun auch Daten sendet, führt dazu, dass das Verschlüsseln der gespeicherten Daten deutlich weniger ins Gewicht fällt. Erklären lässt sich diese Beobachtung dadurch, dass die in diesem Beispiel aufgeführten Module teilweise auf den gleichen Teilen des Betriebssystems aufbauen. So benötigt das verschlüsselte Speichern im Flashspeicher Komponenten, die mit dem CC2420 Funkchip interagieren, um auf dessen Verschlüsselungshardware zugreifen zu können.

		nichts speichern	Klartext speichern	verschlüsselt speichern
nichts übertragen		1.786	8.756	10.438
Klartext übertragen		12.310	17.452	18.236
verschlüsselt übertragen	**CTR**	15.566	20.708	21.164
	CBC-MAC	15.640	20.782	21.238
	CCM	15.644	20.786	21.242

Tabelle 6.4: Vergleich des Speicherbedarfs verschiedener Kombinationen von (un-)verschlüsseltem Senden und (un-)verschlüsseltem Speichern. Alle Werte sind in Byte angegeben.

Aufgrund dieser Ergebnisse eignet sich die Hardwareverschlüsselung des CC2420 Funkchips als Basisschutz, der für jegliche Anwendungen transparent eingesetzt werden kann. Hierbei wird konkret der Einsatz von CCM empfohlen, da der Nutzen (neben der Vertraulichkeit durch den CTR-Modus auch die Integrität durch CBC-MAC, wodurch Replay-Angriffe unterbunden werden) die entstehenden Kosten deutlich überwiegt. Darüber hinaus können Daten, die im Flashspeicher abgelegt werden sollen, ohne merkliche zusätzliche Kosten verschlüsselt werden.

6.4.2 Asymmetrische Kryptographie

Im Gegensatz zur symmetrischen Kryptographie wird bei der asymmetrischen Kryptographie zwischen einem öffentlichen und einem geheimen Schlüssel unterschieden. In Bezug auf Signaturen ist damit sichergestellt, dass nur der Inhaber des geheimen Schlüssels eine Signatur erstellen kann. Überprüft werden kann sie von jeder Instanz, die über den öffentlichen Schlüssel verfügt. Durch die Trennung von öffentlichem und geheimem Schlüssel gestaltet sich auch die Verteilung des Schlüssels deutlich unkomplizierter, da nur die Authentizität des neuen Schlüssels sichergestellt werden muss.

Da für die asymmetrische Kryptographie derzeit keinerlei hardwareseitige Unterstützung auf Sensorknoten verfügbar ist, muss hier auf Softwarelösungen zurückgegriffen werden. Hierzu finden sich in der Literatur vorwiegend Ansätze für die Verschlüsselungsverfahren nach Rivest, Shamir und Adleman (bekannt als RSA-Algorithmus) sowie auf Basis von elliptischen Kurven, Elliptic Curve Cryptography (ECC). Der Hauptunterschied besteht darin, dass sich die Sicherheit des RSA-Algorithmus auf dem sogenannten RSA-Problem beruht, das sich auf das Faktorisierungsproblem zurückführen lässt, d.h. aus dem Produkt zweier sehr großer Primzahlen lassen sich nur schwer die beiden ursprünglichen Primzahlen berechnen. ECC arbeitet hingegen mit einem Zahlenraum, der durch eine elliptische Kurve bestimmt wird. Die Sicherheit des Verfahrens basiert darauf, dass eine bestimmte Operation in diesem Zahlenraum nur sehr schwer invertiert anwendbar ist. Im ursprünglichen Digital Signature Algorithm (DSA) war diese Operation die Exponentiation. Auf dem durch elliptische Kurven definierten Zahlenraum hat man es einer Multiplikation zu tun. Dennoch spricht man

Verfahren	**Schüssellänge**				
ECC	160	224	256	384	512
RSA	1.024	2.048	3.072	7.680	15.360
AES			128	192	256

Tabelle 6.5: Schlüssellängen in Bits für ECC, RSA und AES für jeweils äquivalente Sicherheit. Bewertung durch NIST [13].

allgemeinhin vom Diskreter Logarithmus Problem (DLP), also dem Problem die Umkehrfunktion einer disreketen Exponentiation zu berechnen. Während in diesem Bericht nicht auf die Einzelheiten dieser beiden Algorithmen eingegangen werden soll, sind es jedoch diese Unterschiede, die zu einer abweichenden Bewertung beider Algorithmen führen. Wie in einer Veröffentlichung von National Institute of Standards and Technology (NIST) [13] beschrieben, ist das Maß an Sicherheit beider Verfahren für eine gegebene Schlüssellänge unterschiedlich. Mit anderen Worten, eine Schlüssellänge von 160 Bit für ECC führt zu einer äquivalenten Sicherheit, die durch RSA mit einer Schlüssellänge von 1.024 Bit gegeben ist. Zudem vergrößert sich diese Diskrepanz mit wachsender Schlüssellänge. Tabelle 6.5 gibt einen Überblick über die von NIST vorgeschlagenen Schlüssellängen beider Verfahren, um jeweils eine äquivalente Sicherheit zu gewährleisten. Zum Vergleich wird ebenfalls das symmetrische Verfahren AES aufgeführt.

In einem direkten Vergleich haben Gura et al. [67] bereits 2004 eine Implementierung der Basisoperationen des RSA Algorithmus denen eines ECC Algorithmus gegenübergestellt. Um bspw. eine Signatur gemäß des Elliptic Curve DSA (ECDSA) zu generieren oder zu verifizieren, sind jedoch mehrere Schritte notwendig. Bei äquivalenter Schlüssellänge, siehe Tabelle 6.5, hat sich herausgestellt, dass Verschlüsselungsoperationen mit RSA um mehr als einen Faktor zehn deutlich langsamer sind, das Entschlüsseln aber fast doppelt so schnell erfolgt als die Basisoperation von ECC. Diese Unterschiede verschieben sich jedoch mit zunehmender Schlüssellänge zugunsten ECC. Ähnlich verhält sich ein Vergleich des Speicherbedarfs: die ECC Implementierung benötigt für kleine Schlüssel gut $3,5$-mal so viel Speicher, mit zunehmender Schlüsselgröße schrumpft diese Diskrepanz jedoch. Einen Vergleich verschiedener ECC Implementierungen haben Seo et al. [147] vorgenommen. Dort wurde der Zeitaufwand einer vollständigen Generierung bzw. Verifikation einer Signatur gemessen. Die erzielten Werte unterscheiden sich erwartungsgemäß von den von Gura et al. beschriebenen. Ein Vergleich mit einer RSA Implementierung wurde hier jedoch nicht vorgenommen. Ein vollständiger praxisnaher Vergleich wäre hier wünschenswert, um belastbarere Aussagen treffen zu können.

Von den oben genannten Punkten wiegt vor allem die Schlüssellänge schwer, da hierdurch auch die Länge einer Signatur maßgeblich beeinflusst wird. Bei einer Schlüssellänge von 1.024 Bit entsteht demzufolge eine Signatur derselben Länge, also 128 Byte. Somit passt eine Signatur aufgrund des Overheads durch Header nicht mehr in ein einziges Paket. Die deutlich kleinere Schlüssellänge bei ECC führt dazu, dass auch die Signaturen erheblich kleiner werden.

Implementierung

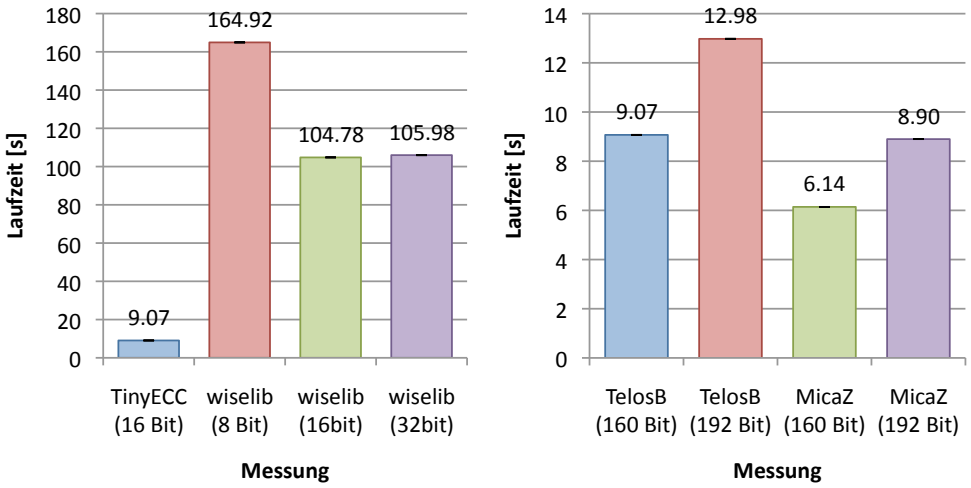
Bei der Implementierung wurde auf vorhandene Bibliotheken zurückgegriffen. Zunächst wurde die Implementierung, die Teil der *Testing*-Version der wiselib [16] ist, von C++ nach C portiert, so dass sie sowohl in TinyOS als auch in Contiki OS verwendet werden kann. Für TinyOS wurde dabei auf die Verwendung des komponenten-basierten Systems verzichtet und stattdessen reiner C-Code eingebunden. Dadurch kann der selbe Quelltext für beide Betriebssysteme verwendet werden, was die spätere Wartung erleichtert. Es werden die Domänenparameter SECP128R1, SECP160R1 und SECP192K1 für elliptische Kurven gemäß den Standards for Efficient Cryptography (SEC) 2 [154] unterstützt. Diese bestimmen durch sechs Parameter u.A. die elliptische Kurve, die wiederrum den Zahlenraum bestimmt.

Im Anschluss wurde eine weitere Implementierung, TinyECC 2.0 [102], hinzugenommen. Sie unterscheidet sich von der wiselib-Version dahingehend, dass deutlich mehr algorithmische Optimierungen implementiert wurden, die signifikante Verbesserungen in der Laufzeit mit sich bringen. Zudem werden mit SECP128R1, SECP128R2, SECP160R1, SECP160R2, SECP160K1, SECP192R1 und SECP192K1 deutlich mehr Domänenparameter unterstützt.

Evaluation

In diesem Abschnitt wird vorwiegend auf die Effizienz der beiden betrachteten Implementierungen eingegangen. Hierfür wurde einem Sensorknoten eine signierte Datei gesendet, der sie dann verifiziert. Die Laufzeit wurde mit einem vom Betriebssystem bereitgestellten Timer auf dem Sensorknoten gemessen und über eine serielle Schnittstelle an einem Computer angezeigt. Jede Konfiguration wurde mit 50 Replikationen gemessen, um anschließend eine statistische Auswertung zu ermöglichen.

In Abbildung 6.10(a) ist eine Vergleichsmessung der beiden betrachteten Implementierungen dargestellt, durchgeführt auf TelosB Knoten. Die kleinen schwarzen Striche an den oberen Enden der Balken stellen den Standardfehler dar, die jedoch aufgrund der geringen gemessenen Varianz sehr klein ausfallen. Es wird sofort deutlich, dass die Implementierung der wiselib *Testing*-Version eine deutlich höhere Laufzeit aufweist, was auf die fehlenden algorithmischen Optimierungen zurückgeführt werden kann. Es wurde zudem überprüft, wie die Ausführungsgeschwindigkeit der wiselib Implementierung mit der Wortlänge, also der Verarbeitungsdatengröße, zusammenhängt. Aus theoretischer Sicht stellt die native Wortlänge eines Prozessors auch die zu verwendende optimale Wortlänge dar. Für TelosB Knoten beträgt die native Wortlänge 16 Bit. Getestet wurden Wortlängen von 8, 16 und 32 Bit. Eine Wortlänge von 8 Bit bedeutet, dass ein 128-Bit-Schlüssel durch einen Array aus 16 Wörtern dargestellt werden kann, wohingegen bei einer Wortlänge von 32 Bit für den gleichen Schlüssel nur 4 Wörter benötigt werden. Während die Byte-Größe des Arrays gleich bleibt, ändert sich für Schleifen, die auf diesen Arrays arbeiten, die Anzahl der Schlei-



(a) wiselib vs. TinyECC

(b) TelosB vs. MicaZ

Abbildung 6.10: Messergebnisse der beiden Bibliotheken für das Verifizieren einer Signatur. Links: Vergleich zwischen wiselib und TinyECC auf einem TelosB Knoten bei gleichbleibender Schlüssellänge von 160 Bit. Die Werte in Klammern geben die Wortlänge an. Rechts: Vergleich zwischen TelosB und MicaZ Knoten mit der TinyECC Implementierung. Die Wortlänge ist für den Knoten jeweils optimal gewählt, die Schlüssellänge ist in Klammern angegeben.

fendurchläufe. Abbildung 6.10(a) zeigt recht anschaulich, dass eine zu kleine Wortlänge einen starken Einfluß auf die Ausführungsgeschwindigkeit haben kann, wenngleich sich die Ausführungsgeschwindigkeit jedoch nicht verdoppelt. Letzteres ist auch nicht zu erwarten, da neben der Verarbeitung großer Zahlen (wie z.B. einem 128-Bit-Schlüssel) auch weitere Operationen durchgeführt werden.

Aufgrund der offensichtlichen Defizite der wiselib-Implementierung wurde für einen direkten Vergleich von TelosB und MicaZ Knoten lediglich TinyECC betrachtet. Abbildung 6.10(b) zeigt die entsprechenden Messergebnisse. Der Prozessor des TelosB Knotens (TI MSP430 [160, 161]) besitzt eine Taktrate von 8 MHz und arbeitet mit einer nativen Wortlänge von 16 Bit. Der Prozessor des MicaZ Knotens (Atmel ATmega128L [11]) ist mit 16 MHz getaktet, arbeitet dafür aber nur mit einer Wortlänge von 8 Bit. Damit lässt sich auch begründen, weshalb die MicaZ Plattform eine Signatur deutlich performanter (mehr als 30%) verifizieren kann als ein TelosB Knoten. Wie oben beschrieben, bewirkt eine Verdoppelung der Schleifendurchläufe keine Verdoppelung der Ausführungszeit. Bezieht man im Weiteren die doppelte Taktrate mit ein, kann ein Geschwindigkeitsvorteil von 30% erklärt werden. Hinzu kommen tiefergehende Details, wie ein unterschiedliches Instruktionsset der Prozessoren, unterschiedlich optimale Assemblerimplementierungen von Subroutinen etc., auf die jedoch im Rahmen des Berichts nicht weiter eingegangen werden kann.

6.5 Sicheres Over-The-Air Programming

Ein adäquater Sicherheitsmechanismus ist für die OTAP-Anwendung eines WSN von essentieller Bedeutung zum Schutz vor unautorisiertem Einspielen bösartiger Software-Aktualisierungen in das Sensornetz. Zur Realisierung eines derartigen Mechanismus ist im Konzept der Sicherheitsarchitektur (siehe Kapitel 5) eine signatur-basierte Kryptographieerweiterung der OTAP-Anwendung vorgesehen. Dabei wird der Programm-Binärcode, der mithilfe der OTAP-Anwendung zur Ausführung an Sensorknoten im Netzwerk übertragen wird, durch eine digitale Signatur geschützt, die sowohl die Integrität als auch die Authentizität des Binärcodes garantieren muss. In Abschnitt 5.6.5 wurde erwähnt, dass sich dazu sowohl symmetrische Verfahren (AES im CBC-MAC-Modus) als auch asymmetrische Verfahren anbieten. Auch die Vor- und Nachteile beider Verfahren wurden dort bereits angesprochen. Der Kernunterschied asymmetrischer Verfahren im Vergleich zu symmetrischen Verfahren ist die Differenzierung zwischen einem öffentlichen und einem geheimen Schlüssel. Mit Hilfe dieser Schlüssel ist gewährleistet, dass ausschließlich der Inhaber des geheimen Schlüssels Signaturen generieren kann, während jede Instanz, die über den öffentlichen Schlüssel verfügt, in der Lage ist, diese Signaturen zu verifizieren ohne den geheimen Schlüssel zu kennen (vergleiche Abschnitt 6.4.2). Daher ist es bei der auf asymmetrischer Kryptographie basierten OTAP-Erweiterung möglich, dass der geheime Schlüssel zur Erzeugung der digitalen Signatur eines Binärcodes ausschließlich dem autorisierten OTAP-Initiator bekannt ist, während alle Sensorknoten lediglich über den dazugehörigen öffentlichen Schlüssel verfügen. Diese Art der Schlüsselverteilung verschafft dem Einsatz von asymmetrischen Verfahren zur Generierung von Signaturen in der sicheren OTAP-Anwendung zwei signifikante Vorteile. Zum einen besteht keine Bedrohung durch Kompromittierung der OTAP-Anwendung bei physikalischer Übernahme eines Sensorknotens durch den Angreifer, da dieser lediglich Zugriff auf den ohnehin bekannten öffentlichen Schlüssel des OTAP-Initiators erlangen kann. Zum anderen ermöglicht ein Propagieren des Programm-Binärcodes via Multicast- bzw. Broadcast-Übertragung ein effizientes gruppen- bzw. netzwerkweites OTAP, denn es existiert ein gemeinsamer (öffentlicher) Schlüssel zur Verifikation der Signatur. Mittels paarweiser symmetrische Kryptographie ist dies hingegen nicht möglich.

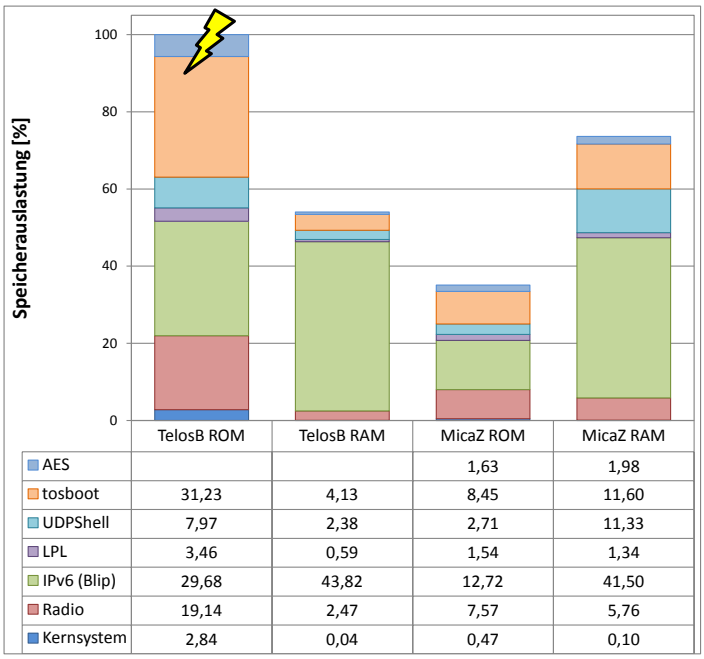
Die im Abschnitt 6.4.2 durchgeführte Performance-Evaluation unterschiedlicher Implementierungen asymmetrischer kryptografischer Verfahren gibt Aufschluss über die Laufzeit bei der Ausführung auf ressourcenbeschränkten Sensorknoten. Abbildung 6.10(b) veranschaulicht die zur Verifikation einer signierten Datei benötigte Zeit unter Verwendung von TinyECC, die in einem Bereich von ca. 10 s liegt und in Abhängigkeit von der Hardware-Plattform und der Schlüsselgröße variiert. Für den Einsatz von asymmetrischer Kryptographie im Rahmen der signatur-basierten OTAP-Anwendung erscheint die TinyECC-Implementierung hinreichend effizient und die zusätzliche Laufzeit tolerierbar, insbesondere im Hinblick auf die damit verbundenen Vorteile gegenüber symmetrischer Verfahren. Aus diesem Grund wurde sich an dieser Stelle für die

Umsetzung eines auf TinyECC-basierten sicheren OTAPs entschieden. Bei der Wahl der geeigneten Schlüsselgröße existiert ein Trade-Off zwischen der erzielten Sicherheit und der zur Verifikation einer Signatur erforderlichen Laufzeit. Die höhere Laufzeit bei einer Schlüsselgröße von 192 Bit im Vergleich zu einer Schlüsselgröße von 160 Bit (siehe Abbildung 6.10(b)) wird im Hinblick auf die gewonnene Sicherheit für den Einsatz im OTAP als geringfügig erachtet. Daher wurde hier die 192 Bit Schlüsselgröße ausgewählt. Die Umsetzung der sicheren OTAP-Anwendung wird im Folgenden beschrieben. Dabei wird auf die potentiellen Bedrohungen eines signatur-basierten Ansatzes eingegangen, die bei der Umsetzung berücksichtigt werden.

6.5.1 Implementierung

In Abschnitt 3.3.2 wurde auf die Problematik der Skalierbarkeit und des hohen Ressourcenbedarfs der ursprünglich für das Contiki- und das TinyOS-Netzwerk vorgesehenen OTAP-Anwendung Deluge [80] eingegangen. Aus dieser Problematik ergab sich die Realisierung einer einfachen OTAP-Basisanwendung, die unter anderem auf den komplexen und aufwendigen Verteilungs-Algorithmus von Deluge verzichtet. Diese Basisanwendung wird nun in einem nächsten Schritt um eine adäquate Sicherheitskomponente erweitert. Dazu wurde in TinyOS nach einer Alternative zu Deluge gesucht und in der Berkeley Low-power IP stack (BLIP)-Erweiterung gefunden. BLIP ist eine Kollektion diverser Internet Protocol (IP)-basierter Protokolle in TinyOS. Darunter befindet sich das Network-Programming-Protokoll *nwprog* [20, Abschnitt 1.5.5], das auf Deluge basiert und den gestellten Anforderungen entspricht. Anhand eines auf der Basisstation ausgeführten Python-Skripts lassen sich Software-Aktualisierungen über einen speziellen Gateway-Knoten in den Flashspeicher von Sensorknoten übertragen und ebenfalls entfernen. Auf den Sensorknoten stellt die *nwprog*-Anwendung eine Konsole (UDPShell) bereit. Via Remote-Zugriff auf diese Konsole kann das Booten der Aktualisierung instruiert werden. Dabei wird der in den Flashspeicher geschriebene Binärcode in den Programmspeicher geladen und anschließend mithilfe des Bootloaders (*tosboot*) ausgeführt.

Bei dem Versuch, die *nwprog*-Anwendung um einen signatur-basierten Sicherheitsmechanismus zu erweitern und in die Sicherheitsarchitektur einzubinden wurde festgestellt, dass der Programmspeicher der eingesetzten Hardware-Plattformen bereits alleine durch diese Anwendung nahezu ausgelastet ist und eine Integration dieser Anwendung in die bestehende Sensorapplikation nicht möglich ist. Schon das Erweitern der Anwendung um den AES-Basisschutz (vgl. Abschnitt 6.4.1) überschreitet den Programmspeicher eines TelosB-Knotens. Abbildung 6.11 veranschaulicht die komponentenweise, prozentuale Speicherauslastung der um AES erweiterten *nwprog*-Kernkomponente. Die beiden fehlenden (und daher geschätzten) Werte des TelosB-ROM- und RAM-Bedarfs sind darauf zurückzuführen, dass dem Entwickler der Speicherbedarf nicht mitgeteilt wird, wenn die Anwendung aufgrund einer Überschreitung des ROM-Speicher nicht kompiliert werden kann.

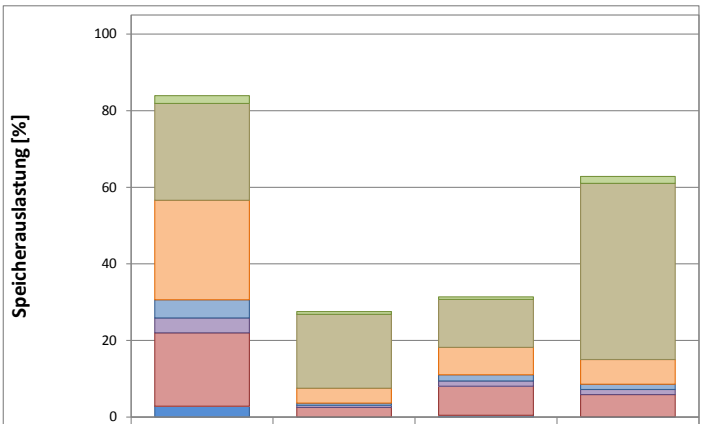


	TelosB ROM	TelosB RAM	MicaZ ROM	MicaZ RAM
AES			1,63	1,98
tosboot	31,23	4,13	8,45	11,60
UDPShell	7,97	2,38	2,71	11,33
LPL	3,46	0,59	1,54	1,34
IPv6 (Blip)	29,68	43,82	12,72	41,50
Radio	19,14	2,47	7,57	5,76
Kernsystem	2,84	0,04	0,47	0,10

Abbildung 6.11: Komponentenweiser Speicherbedarf der in BLIP bereitgestellten OTAP-Anwendung (*nwprog*). Die Grafik zeigt die prozentuale Speicherauslastung von ROM und RAM aller in der Anwendung eingesetzten TinyOS-Komponenten für die TelosB- und die MicaZ-Plattform. Die AES-Komponente wurde dabei explizit hinzugefügt.

Die komponentenweise Unterteilung des in Abbildung 6.11 visualisierten Speicherbedarfs gibt Aufschluss über die Quantität des Bedarfs aller in *nwprog* genutzten Komponenten. Diese unterscheidet sich in der Hardware-Plattform. Auffällig ist, dass vorwiegend zwei der Komponenten für Auslastung des verfügbaren Speichers verantwortlich sind, nämlich der IPv6-Stack von BLIP und der Bootloader tosboot. Da jede OTAP-Anwendung einen derartigen Bootloader erfordert und in TinyOS zu tosboot keine Alternative mit geringerer Anforderung existiert, ist der durch den Bootloader verursachte Speicherbedarf nicht reduzierbar. Des Weiteren ist zu beachten, dass in der Untersuchung des Speicherbedarfs der Bedarf von Sicherheitskomponenten nicht berücksichtigt ist, bei der Implementierung dieser Komponenten jedoch bedacht werden muss. Daher wurde sich in Abstimmung mit dem Projektauftraggeber dazu entschieden, die zu implementierende sichere OTAP-Anwendung separat zu betrachten und nicht in die bestehende Sensorapplikation zu integrieren.

Abgesehen von dem inhärenten Speicherbedarf des Bootloaders lässt sich der Speicherbedarf der OTAP-Anwendung optimieren. Dies ist primär durch den Austausch des speicherintensiven IPv6-Stacks und der UDPShell erreichbar, dessen Funktionalitäten von *nwprog* nur geringfügig in Anspruch genommen werden. Im Rahmen der Umsetzung des sicheren OTAPs wurde diese Optimierung weiter verfolgt. Basierend auf *nwprog* wurde ein sicheres OTAP-Protokoll



	TelosB ROM	TelosB RAM	MicaZ ROM	MicaZ RAM
IDS	2,06	0,72	0,65	1,78
ECC	25,25	19,28	12,51	46,02
tosboot	26,01	3,90	7,18	6,42
AES	4,76	0,55	1,59	1,39
LPL	3,89	0,59	1,41	1,34
Radio	19,14	2,47	7,57	5,76
Kernsystem	2,84	0,04	0,47	0,10

Abbildung 6.12: Komponentenweiser Speicherbedarf der auf *nwprog* basierenden, im Rahmen des Projekts realisierten sicheren OTAP-Anwendung. Die Grafik zeigt die prozentuale Speicherauslastung von ROM und RAM aller in der Anwendung eingesetzten TinyOS-Komponenten für die TelosB- und die MicaZ-Plattform.

(SecOTAP) entwickelt, das analog zu *nwprog* als Modul implementiert wurde und so eine komfortable Integration in jede Applikation ohne die Anpassung sonstiger Module gewährt. Dabei wurde anstelle des IPv6-Stacks der Active Message (AM)-Stack von TinyOS verwendet und gänzlich auf eine Remote-Konsole wie die UDPShell-Komponente verzichtet, um den Speicherbedarf der Anwendung zu reduzieren. Das Instruieren der in *nwprog* über die Konsole übermittelten Aktionen (Boot und Reboot) erfolgt in der entwickelten Anwendung über das Versenden spezieller Request-Pakete, wie es ebenfalls in Deluge gehandhabt wird. Dazu wurde das Nachrichtenformat von *nwprog* übernommen und um diverse Befehle erweitert. Alle Optimierungen erforderten, wie auch die Integration von Sicherheitsmechanismen, sowohl die Überarbeitung der auf den Sensorknoten ausgeführten OTAP-Applikation als auch die Anpassung des Python-Skripts der Basisstation. Die prozentuale Speicherauslastung der entwickelten SecOTAP-Anwendung auf den eingesetzten Hardware-Plattformen ist in Abbildung 6.12 dargestellt. Der Vergleich mit der Abbildung 6.11 verdeutlicht die massive Reduktion des Speicherbedarfs durch die umgesetzten Optimierungen, trotz der erfolgreichen Integration aller Sicherheitskomponenten, die das AES-, das IDS-Modul sowie das ECC-Modul zur Verifikation von Signaturen einschließt. In den folgenden Unterabschnitten wird zunächst das übernommene Konzept von *nwprog* vorgestellt. Anschließend werden darin integrierten

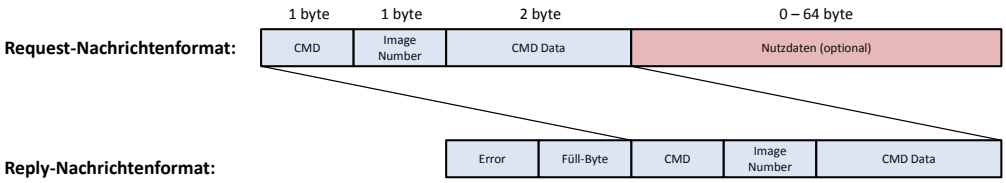


Abbildung 6.13: nwprog-Nachrichtenformat.

Sicherheitsmechanismen und weitere Protokolloptimierungen der entwickelten Anwendung beschrieben.

Konzept des Network Programming Protokolls

Der Programm-Binärcode zur drahtlosen Reprogrammierung in TinyOS ist in der Regel mehrere Kilobytes groß. Folglich ist es nicht möglich, derartigen Code in einem einzigen IEEE 802.15.4 Paket zu übertragen. Stattdessen bedarf es einer Aufteilung des Binärcodes in eine Vielzahl von Paketen, die in der *nwprog*-Anwendung in einem sukzessiven Paketstrom zugestellt werden. In einer *nwprog*-Nachricht steht für den Transfer des Codes ein optionales Nutzdatenfeld mit einer maximalen Größe von 64 Byte zur Verfügung. Diese Größe wurde, sowie das gesamte Nachrichtenformat, in der realisierten SecOTAP-Anwendung übernommen. Das adaptierte Format der zwei in *nwprog* verwendeten Nachrichtentypen ist in Abbildung 6.13 dargestellt sind. Eine Request-Nachricht wird durch das Python-Skript generiert und von der Basisstation an einen Sensorknoten versendet. Dieses Paket enthält einen Befehl (CMD) mit optionaler zusätzlicher Information, die in das CMD Data Feld gespeichert wird. Dieser Befehl wird bei Empfang auf dem Sensorknoten ausgeführt und bezieht sich dabei auf den im Image Number Feld spezifizierten Image-Speicherbereich. In der *nwprog*-Anwendung, sowie auch in der entwickelten Anwendung, sind im Flashspeicher zwei Bereiche zur Speicherung von Binärcode vorgesehen. Damit ist es beispielsweise möglich, dass Knoten auf Veranlassung zwischen zwei gespeicherten Programmen alternieren können, ohne diese erneute übertragen zu müssen. Das optionale Nutzdatenfeld dient der Übertragung von zusätzlich zu dem Befehl benötigter Information, wie zum Beispiel einem Anteil des Binärcodes. Der zweite Nachrichtentyp ist die Reply-Nachricht, die als ACK für jedes eingehende Request von den Sensorknoten versendet wird. Diese Nachricht enthält neben dem Error-Feld, das den Erfolg bzw. das Scheitern des im Request-Paket übermittelten Befehls signalisiert, die gleichen Felder wie die Request-Nachricht ohne das optionale Nutzdatenfeld. In diese Felder wird der Inhalt des Request-Pakets kopiert, welches das Reply-Paket ausgelöst hat. Dies dient der korrekten Zuordnung des Requests in der Basisstation.

Im Nachrichtenformat von *nwprog* existieren keine Sequenznummern zur Duplikatunterscheidung. Prinzipiell könnten für WRITE- und READ-Pakete zu diesem Zweck die eindeutige Kombination aus der Image Number und der im CMD Data Feld enthaltenen Speicherposition genutzt werden. Der Request-Reply-Mechanismus ist durch eine Stop-and-Wait Strategie implementiert, d.h.

CMD	**Img. Nr.**	**CMD Data**	**Nutz-daten**	**Erklärung:**
ERASE	x	–	–	Löschen des in Image Number spezifizierten Speicherbereichs x.
WRITE	x	*Position*	*Daten*	Schreiben der Daten in die entsprechende Position des Speicherbereichs x.
READ	x	*Position*	–	Lesen der Daten aus der entsprechenden Position des Speicherbereichs x.
BOOT	x	Zeit	–	Ausführen des im Speicherbereich x enthaltenen Binärcodes mit einer festgelegten Verzögerung.
REBOOT	–	Zeit	–	Reboot des Sensorknotens nach einer festgelegten Verzögerung.
WRITE_ BEGIN	x	Counter	–	Initiierung eines Prozesses zum Schreiben des Binärcode mit vordefiniertem Counter in den Speicherbereich x.
WRITE_ END	x	–	Signatur	Beenden eines Schreibprozesses in den Speicherbereich x und gleichzeitiger Übertragung der Signatur zur Verifikation des übertragenen Binärcodes.

Tabelle 6.6: OTAP-Befehle und zugehörige Request-Pakete in der SecOTAP-Anwendung. Das Zeichen – symbolisiert dabei, dass das entsprechende Datenfeld aus dem Header des Request-Paket für den jeweiligen Befehl nicht genutzt wird.

nach jedem Request-Versand wird auf das dazugehörige Reply-Paket gewartet, bevor die Datenübertragung fortgeführt wird. Dadurch sind Reihenfolgevertauschungen bei der Paketübertragung ausgeschlossen. Das Warten auf ein ausgebliebenes Reply-Paket wird durch einen Timeout beendet. Ist ein solcher Timeout aufgetreten oder signalisiert das Error-Feld des Reply-Pakets einen Fehler, wird das Request-Paket erneut gesendet (standardmäßig bis zu zwei Mal).

In der Tabelle 6.6 sind die drei im originalen *nwprog*-Protokoll definierten Befehle, ERASE, WRITE und READ (kursiv), zusammen mit der zugehörigen Belegung des Request-Nachrichtenformats aufgelistet. Ferner sind zudem die in der entwickelten SecOTAP-Anwendung definierten Befehle in die Tabelle aufgenommen. Dies sind zum einen die bereits angesprochenen Befehle BOOT und REBOOT, zum anderen zwei Befehle, die Teil der Sicherheitsmechanismen sind, auf die im nachfolgenden Unterabschnitt eingegangen wird.

Sicherheitsmechanismen der SecOTAP-Anwendung

Die umgesetzte OTAP-Anwendung wurde wie eingangs erläutert durch das TinyECC-Modul mit einer Schlüsselgröße von 192 Bit zu der signatur-basierten SecOTAP-Anwendung erweitert. Die optionale AES-Verschlüsselung der An-

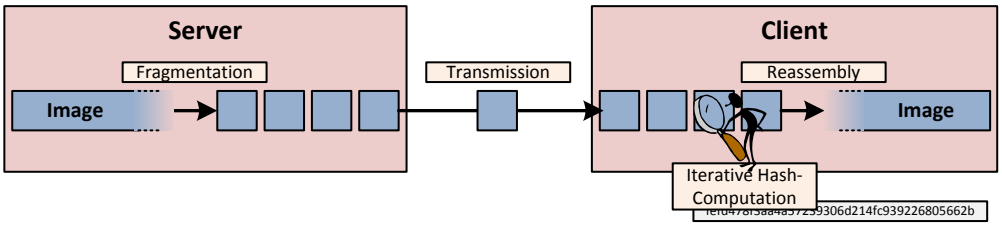


Abbildung 6.14: Paketweise Übertragung und iterativer Hash-Berechnung eines Programm-Binärcodes.

wendung bildet dabei den Basisschutz gegen Insider-Angriffe gegen die OTAP-Anwendung.

Das aufwendige Verifizieren einer Signatur kann aus Performance- und Effizienzgründen nicht paketweise durchgeführt werden, da die Laufzeit der Berechnung für jedes Paket zu groß wäre (vgl. Abschnitt 6.4.2). Stattdessen wird lediglich der Programm-Binärcode bzw. dessen SHA-1-Hash-Wert signiert. Auf dem zu reprogrammierenden Sensorknoten wird anhand der Signatur die Integrität und die Authentizität eines Binärcodes überprüft, sobald dieser vollständig übertragen wurde. Dies geschieht, indem der Empfänger nach der Initialisierung eines Schreib-Prozesses (WRITE_BEGIN-Paket) bei jedem eingehenden WRITE-Paket eine iterative Hash-Berechnung durchführt, wie in Abbildung 6.14 veranschaulicht. Während dieser Berechnung ist eine Vertauschung der Reihenfolge eingehender WRITE-Pakete nicht tolerabel und durch die Stop-and-Wait Übertragungsstrategie ausgeschlossen. Das Ergebnis der Hash-Berechnung wird nach vollständigem Erhalt des Programm-Binärcodes anschließend mit der im WRITE_END-Paket enthaltenen Signatur verifiziert. Scheitert die Verifikation, stammt der empfangene Binärcode entweder nicht von dem autorisierten OTAP-Initiator oder der Binärcode wurde während der Übertragung gezielt durch fremde Daten manipuliert. In beiden Fällen handelt es sich um einen OTAP-spezifischen Insider-Angriff, auf den durch das unmittelbare Entfernen des empfangenen Codes aus dem Flashspeicher und durch Senden einer IDS-Alarmnachricht reagiert wird. Zudem sperrt das SecOTAP-Modul den Sensorknoten für einen festgelegten Zeitraum (aktuell 5 min). In diesem Zeitraum ist die SecOTAP-Anwendung auf dem Sensorknoten deaktiviert. Diese Sperrung dient als Gegenmaßnahme gegen möglicherweise weiter folgende Angriffe.

Ferner besteht die Gefahr von Reply-Attacken. Im Kontext von OTAP ergibt sich diese Form des Angriffs durch das Wiedereinspielen eines mitgehörten und eventuell veralteten Programm-Binärcodes an möglicherweise zusätzliche OTAP-Empfänger. Zum Schutz gegen derartige Angriffe wurde ein Versionszähler (Counter) für den Binärcode hinzugefügt. Der Counter ist vom OTAP-Initiator bei dem Propagieren jeder Aktualisierung stets zu inkrementieren. Des Weiteren ist in dem SecOTAP-Modul das Initialisieren eines Schreib-Prozesses auf den Empfang eines WRITE_BEGIN-Pakets mit höherem Counter beschränkt, so dass veraltete Binärcodes verworfen werden. Gleichzeit wird ein spezieller IDS-Alarm an den IDS-Server übertragen, der den Replay-Angriff mitteilt. Um die Integrität des Counters und ebenfalls die Integrität der Empfänge-

IDS-Ereignis:	**Detektionspotential:**
Fehlgeschlagene Verifizierung der Signatur	Insider- bzw. Replay-Angriffe
WRITE_BEGIN mit invalidem Counter	Insider- bzw. Replay-Angriffe
Empfang von unerlaubtem BOOT (SECURE_MODE)	Insider-Angriffe
Empfang von unerlaubtem REBOOT (SECURE_MODE)	Insider-Angriffe
Timeout während Schreib-Prozess	DoS-Angriffe (z.B. Jamming-Angriff)
Erneutes WRITE_BEGIN während laufendem Schreib-Prozess	Insider-Angriffe
(Re-)Boot eines Sensorknotens	Insider-Angriffe (falls unautorisiertes Booten)

Tabelle 6.7: IDS-Alarm auslösende Ereignisse und deren Detektionspotential.

radresse zu gewährleisten, wurde die Domäne der Signatur auf den Counter und die Empfängeradresse erweitert.

Ein optionaler Modus (SECURE_MODE) bietet über den bestehenden Sicherheitsmechanismen hinaus die Option, das Booten eines Binärcodes nur im direkten Anschluss an eine erfolgreiche Signaturverifikation zu erlauben. Erhält ein Knoten zu einem sonstigen Zeitpunkt ein BOOT-Paket, wird diesem Knoten das Booten für ein vordefiniertes Zeitintervall von 10 s untersagt. Das Rebooten von Sensorknoten wird dagegen permanent deaktiviert. Beide Mechanismen dienen als Gegenmaßnahme gegen Insider-Angriffe und lösen ebenfalls eine IDS-Benachrichtigung aus.

Eine Gesamtübersicht über alle von der SecOTAP-Anwendung generierbaren IDS-Nachrichten ist in der Tabelle 6.7 gegeben. Neben den bisher erwähnten Alarmtypen existieren drei weitere Typen. Zwei davon betreffen einen begonnenen Schreib-Prozess. Auf der einen Seite wird die Unterbrechung des Schreib-Prozesses, die durch einen Timeout erkannte wird und beispielsweise durch einen Jamming- oder Rushing-Angriff (vgl. Abschnitt 6.3) ausgelöst wurde, mittels einer IDS-Nachricht signalisiert. Auf der anderen Seite signalisiert eine weitere IDS-Nachricht, den Versuch eines Angreifers, den laufenden Schreib-Prozess durch ein erneutes WRITE_BEGIN-Paket zu unterbrechen. Gegen diese Art von Angriffen ist der Schreib-Prozess geschützt, indem der Versuch eines erneuten Schreib-Vorgangs während des laufenden Prozesses pauschal ignoriert wird. Der zuletzt aufgeführte IDS-Alarm wird bei jedem Booten bzw. Rebooten eines Sensorknotens verschickt und weist den Sensornetzbetreiber auf ein möglicherweise unautorisiertes Booten des Sensorknoten hin.

Selektives OTAP im Sensornetz

Das ursprüngliche *nwprog*-Protokoll ist für das separate Reprogrammieren von Sensorknoten via Unicast-Transfer ausgelegt. Ein selektives Reprogrammieren einer Teilmenge aller Sensorknoten im Netzwerk ist hier

– wenn auch vom Python-Skript unterstützt – lediglich durch sukzessive Unicast-Reprogrammierung möglich. Im Gegensatz dazu wurde die SecOTAP-Anwendung um eine optionale Multi- und Broadcast-Unterstützung erweitert, die ein effizientes gruppen- bzw. netzwerkweites OTAP erlaubt. Der Multicast-Modus ist dabei durch einen Explicit Multicast (Xcast)-Ansatz [22] realisiert. Das bedeutet, dass die beabsichtigten Empfänger eines Pakets anhand einer expliziten Liste von Empfängeradressen in dem Paket-Header festgelegt sind, während das Paket an die Broadcast-Adresse adressiert ist. Zur Unterscheidung zwischen einer Broadcast-Übertragung wird im Broadcast-Modus hingegen die fixe Broadcast-Adresse `0xFFFF` als alleinige Adresse in der Empfängerliste angegeben. Zur Speicherung der Liste wird das optionale Nutzdatenfeld des originalen *nwprog*-Nachrichtenformat (siehe Abbildung 6.13) genutzt. Ausnahmen bilden dabei WRITE- und WRITE_END-Pakete, deren Nutzdatenfeld bereits durch die in den Flashspeicher zu schreibenden Daten bzw. durch die Signatur belegt ist (vergleiche Tabelle 6.6). Bei beiden Pakettypen ist die Empfängerliste jedoch nicht zwingend erforderlich, da aus Sicherheitsgründen ausschließlich Knoten auf diese Pakete reagieren, die zuvor aufgrund eines eingehenden WRITE_BEGIN-Pakets einen Schreibvorgang initialisiert haben. Wie auch im Falle der Unicast-Übertragung im vorigen Unterabschnitt erzeugt die Basisstation als Gegenmaßnahme gegen Replay-Angriffe eine Signatur des Hashs der Konkatenation des Binärcodes mit dem Counter und der Empfängeradresse (hier die Empfängerliste).

Während im Unicast-Modus je ein einziger Knoten ein Reply-Paket generiert, erstellen im Multicast- und im Broadcast-Modus im Allgemeinen alle Empfänger ein derartiges Paket und versuchen, dieses gleichzeitig zu versenden. Daher entsteht im Multi- bzw. Broadcast-Modus eine Konkurrenzsituation um den Medienzugriff zum Versenden der Reply-Pakete. Aus dieser Konkurrenzsituation resultiert eine höhere Kollisionswahrscheinlichkeit bzw. ein größerer Paketverlust der Reply-Pakete und eine größere und variierende Verzögerung beim Versenden der Reply-Pakete. Insbesondere wegen der Varianz des Zeitpunktes des Reply-Versands erfordert die Multi- und Broadcast-Erweiterung des SecOTAP-Moduls eine weitere Anpassungen des zum Versenden von OTAP-Nachrichten genutzten Python-Skripts. Wegen der eingesetzten Stop-and-Wait Strategie beim Versenden des Request-Paketstroms, muss die Basisstation den Empfang sämtlicher Reply-Pakete abwarten, bevor mit der Übertragung des nachfolgenden Request-Pakets fortgefahren werden kann. Im Broadcast-Modus ist jedoch die Anzahl von Broadcast-Empfängern und daher auch die Anzahl zu erwartender Reply-Pakete im Allgemeinen unbekannt. Aus diesem Grund muss im Broadcast-Modus ein künstlicher Zeitpuffer eingeführt werden, der nach Versand jedes Request-Pakets abzuwarten ist, um alle potentiellen Reply-Pakete zu empfangen. Die adäquate Länge dieses Zeitpuffers hängt von der Anzahl zu erwartender Reply-Pakete ab und ergibt sich aus dem Produkt der Anzahl von Broadcast-Empfängern und einer Wartezeit pro Empfänger. In einer Messreihe wurde in Schritten von 5 ms die erforderliche Wartezeit pro Empfänger ermittelt. Diese beträgt ca. 30 ms. Damit in einer konkreten Anwendung ein adäquater Zeitpuffer bestimmt werden kann, ist also die maximale Anzahl von

Empfängern durch den Endnutzer festzulegen bzw. zu schätzen und mit 30 ms zu multiplizieren. Im Multicast-Modus ist die Anzahl der Empfänger hingegen bekannt. Daher ist das Abwarten aller auf ein Reply-Paket folgender Request-Pakete in diesem Modus unproblematisch. Dennoch haben Testläufe im Rahmen der Entwicklung gezeigt, das auch hier ein geringer zusätzlicher Zeitpuffer zwischen dem Empfang der Request-Pakete und dem nachfolgenden Reply-Paket sinnvoll ist. Aktuell wird dazu eine pauschale Dauer von 30 ms verwendet (unabhängig von der Anzahl Multicast-Empfänger).

Des Weiteren ist zu beachten, dass der Reply-Mechanismus des SecOTAP-Protokolls im Broadcast-Modus nicht die vollständige bzw. lückenlose Übertragung des Binärcodes garantieren kann, denn die Übertragung wird kontinuierlich fortgeführt, solange wenigstens ein Reply pro verschicktes Paket bei der Basisstation eingeht. Daher besteht die Möglichkeit, dass im Broadcast-Modus eine invalide Signaturprüfung auch aus einem unvollständig empfangenem Code resultieren kann (Verletzung der Integrität). Zur Garantie einer fehlerfreien Übertragung ist das Einführen von Sequenznummern eine notwendige Voraussetzung, denn nur mithilfe von Sequenznummern kann ein Empfänger den Verlust einzelner Pakete aus dem Paketstrom bemerken und bei Bedarf erneut anfragen.

Der von der realisierten selektiven OTAP-Erweiterung erwartete Effizienzgewinn, insbesondere durch den Multicast-Modus, wird im Kontext der Evaluation der SecOTAP-Anwendung im folgenden Abschnitt näher untersucht.

6.5.2 Evaluation

Zur Evaluation des SecOTAP-Moduls wurde das heterogene TinyOS-Netz des WSN-Testlabors verwendet. Dieses Sensornetz besteht wie in Abschnitt 3.3 beschrieben aus je fünf TelosB- und MicaZ-Knoten. Als Gateway zwischen dem Sensornetz und der Basisstation, die das OTAP initiiert, wurde ein TelosB-Knoten über die UART-Schnittstelle mit der Basisstation verbunden. Für die Evaluation wurde eine einfache Testapplikation implementiert und zu Beginn per UART auf allen Sensorknoten installiert. Die Aufgabe der Testapplikation ist lediglich das Signalisieren der Funktion dieser Applikation durch Aufblinken einer LED. Zudem ist sowohl das SecOTAP-Modul als auch das generische IDS-Modul in die Testapplikation integriert. Damit während der Evaluation zwischen der vorherigen und einer erfolgreich per OTAP installierten, nachfolgenden Version der Testapplikation differenziert werden kann, wurden im Vorfeld zwei Versionen der Testapplikation kompiliert. Diese Versionen verwenden unterschiedliche LEDs und unterscheiden sich so in der Farbe des Aufblinkens. Abbildung 6.15 skizziert exemplarisch die bei der Evaluation durchgeführte Reprogrammierung eines Sensorknotens. In einer Code-Übertragungsphase zwischen dem WRITE_BEGIN- und dem WRITE_END-Paket wird der Binärcode der neuen Testapplikation (grüne LED) an den Empfänger übertragen, auf dem aktuell die ursprüngliche Testapplikation (rotes LED) ausgeführt wird. Ist diese Phase abgeschlossen, beginnt die Verifikation der Signatur. Bestätigt die Signa-

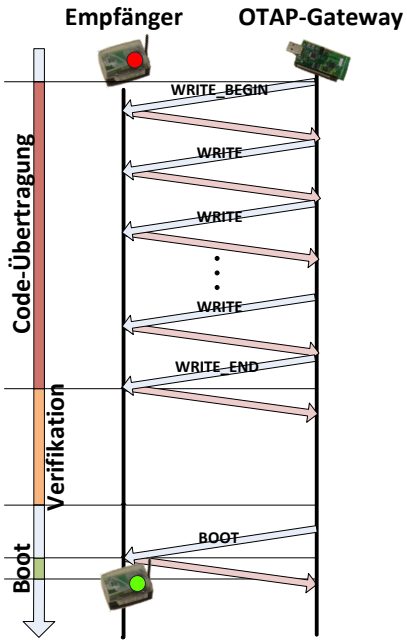


Abbildung 6.15: Pakettransfer und Phasen bei einem erfolgreichen OTAP-Prozess. Nach der vollständigen Übertragung des Binärcodes, wird die empfangene Signatur verifiziert und der Code ausgeführt, sobald ein entsprechendes BOOT-Paket eingeht.

tur die Integrität und Authentizität des empfangenen Codes, wird dieser in der Boot-Phase in den Programmspeicher geladen und ausgeführt, sobald ein BOOT-Paket eingeht. Der Farbwechsel des Aufblinkens bestätigt die erfolgreiche Reprogrammierung. Scheitert dagegen die Verifikation der Signatur, wird der entsprechende IDS-Alarm verschickt.

Zur Bewertung der Performance des sicheren OTAPs werden die Erfolgswahrscheinlichkeit und das bei dem OTAP-Prozess entstandene Datenaufkommen betrachtet, sowie die Dauer dieses Prozesses. Während der Erfolg einer OTAP-Operation anhand des Farbwechsels des Aufblinkens bestimmt werden kann, wird bezüglich beider letztgenannten Metriken der Paket-Sniffer eingesetzt. Der in Wireshark aufgezeichnete Datenverkehr erlaubt eine exakte Bestimmung der Ergebnisse. Das entstandene Datenaufkommen wird als Maß für die Energieeffizienz des sicheren OTAPs betrachtet. Es wird zwischen dem Datenaufkommen aus der Übertragung von Programm-Paketen und aus der Übertragung von Kontroll-Paketen differenziert. Kontroll-Pakete umfassen dabei Reply-Pakete sowie sämtliche Pakete, die keinen Anteil des Programm-Binärcodes transportieren. Beide Datenaufkommen werden durch Addition der Paketgrößen der entsprechenden Pakete ermittelt. Dazu zählen ebenfalls Pakete, die aufgrund von Bitfehlern nicht korrekt empfangen werden können. Die Dauer des OTAP-Prozesses ist hingegen definiert als Zeitspanne zwischen dem initialen WRITE_BEGIN-Paket und dem Empfang des zum abschließenden WRITE_END-Paket gehörenden letzten Reply-Pakets. Zu beachten ist, dass die Zeit, die zur Verifikation der Authentizität des übertragenen Binärcodes er-

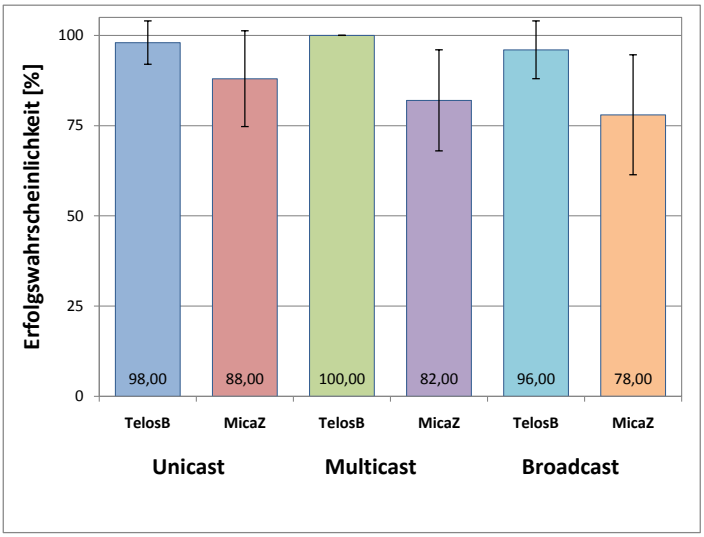


Abbildung 6.16: Erfolgswahrscheinlichkeit verschiedener Modi bei der drahtlosen Reprogrammierung von je fünf Sensorknoten gleichen Typs.

forderlich ist (12,98 s auf der TelosB- und 8,90 s auf der MicaZ-Plattform, vgl. Abschnitt 6.4.2), nicht in diese Dauer einbezogen wurde. Stattdessen handelt es sich hier um die reine Übertragungszeit des Programms. Es wurden drei Messreihen mit je zehn Replikationen durchgeführt, in denen für das OTAP der Unicast-, der (explicit) Multicast- und der Broadcast-Modus verwendet wurde. Da sich der kompilierte Binärcode der Testapplikation für die TelosB- und die MicaZ-Plattform unterscheidet und auf der jeweils anderen Plattform nicht lauffähig ist, wurden die drei Messreihen in je zwei separate Messreihen aufgeteilt. Dazu wurden beide Plattformen bei der Performance-Messung gruppiert. Das bedeutet, dass bei der Unicast-Messreihe die Gruppe der fünf TelosB- bzw. der fünf MicaZ-Knoten per sukzessiver Unicast-Übertragung (also ein Knoten nach dem anderen) reprogrammiert werden. Dagegen wird der Binärcode bei der Multicast-Messreihe parallel per Xcast an die jeweilige Gruppe propagiert. Bei der Broadcast-Messreihe wird analog zur Multicast-Messreihe verfahren, mit dem Unterschied, dass zur Zeit der Übertragung ausschließlich die entsprechende Empfängergruppe des Binärcodes eingeschaltet wurde. Die Ergebnisse der insgesamt sechs Messreihen bezüglich der oben genannten Metriken (inklusive der Varianz in Form der Standardfehler am oberen Ende der Balken) sind in den Abbildungen 6.16, 6.17 und 6.18 visualisiert und werden im Folgenden näher erläutert.

Die Performance-Evaluation der SecOTAP-Anwendung zeigt, dass die Erfolgswahrscheinlichkeit einer OTAP-Operation in allen betrachteten Modi für beide Hardware-Plattformen, TelosB und MicaZ, im Mittel mehr als 75% beträgt (siehe Abbildung 6.16). Auffällig ist jedoch, dass die mittlere Erfolgswahrscheinlichkeit von der eingesetzten Zielplattform abhängt. Die Abbildung verdeutlicht, dass in den Messreihen die Erfolgswahrscheinlichkeit für das Reprogrammieren der TelosB-Knoten durchschnittlich um mindestens 10% höher ist als die

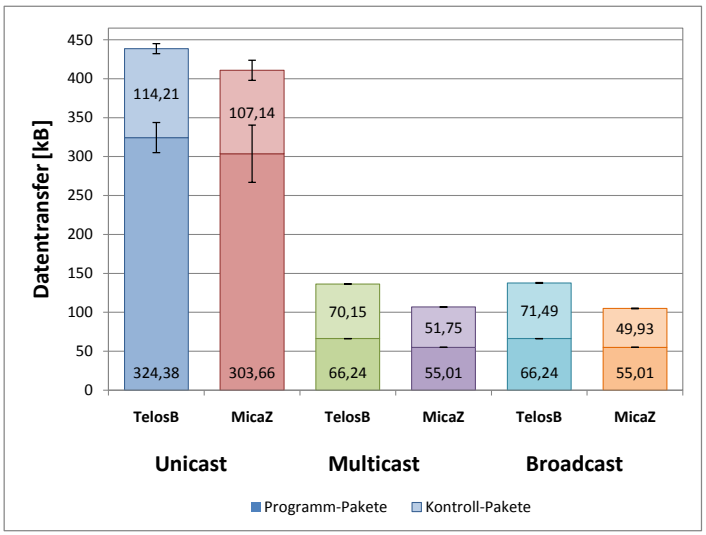


Abbildung 6.17: Datenaufkommen einer OTAP-Operation in verschiedenen Modi bei der drahtlosen Reprogrammierung von je fünf Sensorknoten gleichen Typs.

Wahrscheinlichkeit einer erfolgreichen MicaZ-Reprogrammierung. Die Ursache für diese Diskrepanz konnte im Rahmen des Projekts jedoch nicht eindeutig festgestellt werden. Da beide Plattformen den identischen Funkchipsatz verwenden, ist es denkbar, dass die Ursache in dem unterschiedlichen Flashspeicher der beiden Plattformen begründet ist bzw. der TinyOS-Schnittstelle zwischen diesem Speicher und dem Kernbetriebsystem. Neben der signifikant geringeren mittleren Erfolgswahrscheinlichkeit der Reprogrammierung der Gruppe von MicaZ-Knoten ist zudem eine höhere Varianz der Wahrscheinlichkeit zu erkennen.

Abbildung 6.17 visualisiert das mittlere Datenaufkommen während eines OTAP-Prozesses für unterschiedliche Übertragungsmodi und Zielplattformen. Die Abbildung verdeutlicht die signifikante Reduktion des Datenaufkommens des Multicast- und des Broadcast-Modus gegenüber der seriellen Übertragung des Binärcodes im Unicast-Modus. Erwartungsgemäß wird das Volumen der zum Transport des Binärcodes benötigten Programm-Pakete ca. um den Faktor 5 reduziert, wenn der Binärcode der Testapplikation im Multicast- bzw. im Broadcast-Modus gemeinsam an die fünf Knoten große Empfängergruppe versendet wird. Die leichte Abweichung von dem Faktor 5 ergibt sich aus der Korrelation des Datenaufkommens mit der Erfolgswahrscheinlichkeit im Unicast-Modus. Sobald im Unicast-Modus anhand des Reply-Mechanismus ein Scheitern einer Übertragung registriert wird, wird die Übertragung vorzeitig abgebrochen. Der Broadcast-Modus ist dagegen unabhängig von dem Erfolg der Übertragung. Dies trifft ebenfalls für den Multicast-Modus zu, solange der Erfolg der Übertragung jedes Pakets von mindestens einem Empfänger per Reply bestätigt wird. In jeder Replikation der Messreihen wurde diese Bedingung erfüllt, weshalb das entstandene Datenaufkommen des Multicast- und des Broadcast-Modus identisch ist. Die Differenz des Datenaufkommens zwischen der Gruppe der TelosB- und der Gruppe der MicaZ-Knoten ist durch die

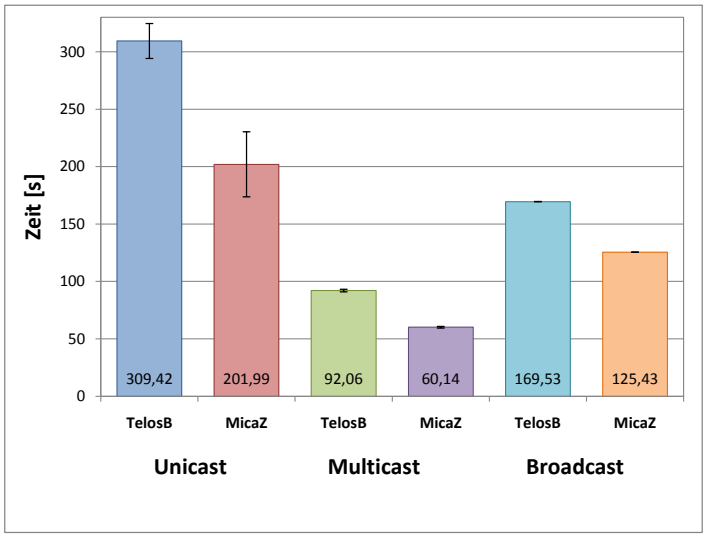


Abbildung 6.18: Dauer einer OTAP-Operation in verschiedenen Modi bei der drahtlosen Reprogrammierung von je fünf Sensorknoten gleichen Typs.

Differenz in der Größe des Binärcodes in Abhängigkeit von der Zielplattform begründet. Diese beträgt für die TelosB-Plattform 45.648 Byte und für einen MicaZ-Knoten hingegen 37.920 Byte. Die Empfangsbestätigung jedes OTAP-Pakets ist wie bei der Unicast-Übertragung auch im Multicast- und Broadcast-Modus vorgesehen. Folglich entsteht unabhängig vom Übertragungsmodus ein vergleichbares Datenaufkommen des Reply-Mechanismus. Daher verringert sich der Faktor der Reduktion unter Berücksichtigung der Kontroll-Pakete. Jedoch zeigt die Evaluation, dass sowohl der Multicast- als auch der Broadcast-Modus beim Versenden der Reply-Pakete zu einer Konkurrenzsituation beim Medienzugriff führt, was das geringere Volumen der Kontroll-Pakete im Vergleich zum Unicast-Modus erklärt.

Die mittlere Dauer einer OTAP-Operation in den betrachteten Übertragungsmodi ist in Abbildung 6.18 dargestellt. Analog zum mittleren Datenaufkommen in Abbildung 6.17 ist auch hier die mit der Erfolgswahrscheinlichkeit korrelierende Varianz des Unicast-Modus, die von der Zielplattform abhängige Differenz und eine signifikante Reduktion durch den Multicast- und den Broadcast-Modus zu beobachten. Der theoretisch mögliche Reduktionsfaktor 5 wird mittels Multicast- bzw. Broadcast-Modus jedoch wegen des in Abschnitt 6.5.1 beschrieben künstlich hinzugefügten Zeitpuffers nicht erreicht. Im Multicast-Modus beträgt dieser stets 30 ms. In den Broadcast-Messungen wurde bezüglich des Zeitpuffers der optimale Fall angenommen und die tatsächliche Anzahl der vorhandenen Empfänger zur Bestimmung des Zeitpuffers verwendet, woraus sich ein Puffer von $5 \cdot 30\,\text{ms} = 150\,\text{ms}$ ergibt. Die in beiden Modi unterschiedlichen Zeitpuffer implizieren eine Verlängerung der OTAP-Operation, aus der die in der Abbildung zu beobachtenden Differenzen zwischen der Multicast- und der Broadcast-Übertragung resultieren. Eine weitere Beobachtung der Betrachtung der mittleren Dauer einer OTAP-Operation ist, dass die zur Verifikation der

Authentizität des Binärcodes notwendige Zeit um ein Vielfaches geringer ist als die zur Übertragung erforderliche Zeit.

Bezüglich dem Vergleich der Übertragungs-Modi lässt sich anhand der drei bei der Evaluation in Betracht gezogenen Metriken abschließend folgendes Gesamtfazit ziehen. Zur selektiven drahtlosen Reprogrammierung mehrerer Sensorknoten ist das Verwenden des (explicit) Multicast- und des Broadcast-Modus aufgrund der besseren Energieeffizienz und der signifikanten Reduktion der benötigten Dauer von erheblichem Vorteil gegenüber dem Unicast-Modus. Dieses Resultat rechtfertigt zudem nachhaltig den Entschluss, asymmetrische Verfahren zur Signaturerzeugung einzusetzen, denn mittels paarweiser symmetrischer Kryptographie ist es nicht möglich, eine gemeinsame valide Signatur an mehrere Empfänger zu senden.

6.5.3 Ausblick

Die Evaluation der Erfolgswahrscheinlichkeit der drahtlosen Reprogrammierung durch die umgesetzte sicherere OTAP-Anwendung hat gezeigt, dass zumindest unter Laborbedingungen auch ohne den Einsatz von Sequenznummern kein wesentlicher Unterschied zwischen der Robustheit des Unicast-, des Multicast- und des Broadcast-Modus zu beobachten ist. Dennoch erscheint die Integration von Sequenznummern zur Steigerung der Robustheit insbesondere für den Praxiseinsatz sinnvoll. Zudem ermöglicht die Integration von Sequenznummern den Einsatz von Fenstermechanismen und NACKs, mit denen unnötige ACKs eingespart werden könnten. Weitere Untersuchung in diesem Bereich erscheinen aus diesem Grund lohnenswert. Darüber hinaus besteht weiterer Forschungsbedarf trotz Speicherrestriktionen auch noch Routing-Funktionalitäten zu integrieren, um auch ein multihop-fähiges sicheres OTAP zu realisieren.

6.6 Simulative Evaluation der Skalierbarkeit

Die Anzahl der Sensorknoten, die in der im Rahmen des Projekts realisierten Testumgebung eingesetzt werden, ist aktuell auf zehn Sensorknoten pro Teilnetz begrenzt. Aus diesem Grund wurde zur Bewertung der Skalierbarkeit der realisierten Sensorapplikation auf Simulationen zurückgegriffen. Die bei diesen Simulationen verwendeten Simulationsumgebungen, die ausgewählten Szenarien, sowie die Ergebnisse dieser Untersuchung werden im Folgenden vorgestellt.

6.6.1 Simulationsumgebung

Im Bereich drahtloser Netzwerke existiert eine Vielzahl von Netzwerksimulatoren, wie beispielsweise der häufig verwendete Simulator Ns-2 [162]. Prinzipiell ist der Einsatz dieser Simulatoren zur Simulation von drahtlosen Sensornetzen möglich. Zur Durchführung der Skalierbarkeitsevaluation erscheinen jedoch

spezielle, zum Testen, Validieren und Evaluieren von Sensornetzprotokollen und Applikationen entwickelte WSN-Simulationsumgebungen als geeigneter. Diese Simulatoren besitzen den Vorteil, dass sich die Sensorapplikation der Knoten, d.h. der entwickelte Programmcode inklusive aller benötigten OS-Komponenten bzw. die für die eingesetzte Hardware kompilierte Binärdatei vollständig und direkt in den WSN-Simulator einbinden lässt. Auf diese Weise muss der entwickelte Programmcode nicht nachträglich in den Simulator portiert werden. Dies ermöglicht ein realistischeres Simulieren. Zudem werden Abweichungen und Fehler ausgeschlossen, die sich aus unterschiedlichen Rechnerarchitekturen und potentiellen Implementierungsunterschieden ergeben können.

Für jedes der im Rahmen des Projekt in Betracht gezogenen WSN-OSs ist ein derartiger Simulator verfügbar. Der Simulator von Contiki nennt sich Cooja [122] und bietet anhand des Emulators MSPSim [52] die Option, jeglichen für TelosB-Knoten kompilierten Binärcode zu emulieren. Der Simulator von iSense, dessen Konzept in Abschnitt 8.3.1 vorgestellt wird, heißt Shawn [56]. Für TinyOS wird der diskrete, ereignisbasierte Simulator TOSSIM [100] angeboten. Aufgrund der eingangs erwähnten Vorteile dieser WSN-Simulatoren gegenüber allgemeinen Netzwerksimulatoren, wurden diese für die durchzuführenden Skalierbarkeitsevaluation ausgewählt, d.h. der Programmcode der Sensorapplikation jedes Betriebssystems wurde in der entsprechenden Simulationsumgebung analysiert.

6.6.2 Szenarien

In den Simulationen wurden vier Szenarien mit statischen Topologien betrachtet, in denen je 100, 500, 1000 und 2000 Sensorknoten auf einer Fläche von $300\,\mathrm{m}^2$ randomisiert (mit Gleichverteilung) positioniert wurden. Als Kommunikationsreichweite der simulierten Funkchips wurde eine Distanz von 50 m gewählt. Diese Distanz stellt eine typische Reichweite von IEEE 802.15.4 Funkchips dar. Auf den Knoten wird die Sensorknotenkomponente der in Abschnitt 3.3.1 vorgestellten Sensorapplikation simuliert, in der jeder Knoten periodisch (alle 5 s) Sensordaten erhebt. In der Simulation werden anstelle der gemessenen Sensordaten konstante Werte verwendet, die dann analog zur realen Applikation über ein baumbasiertes Routing-Protokoll per Anycast an eine Datensenke übermittelt werden. Als Datensenke dient eine Basisstation, die in der Simulation zusätzlich zu der oben genannten festgelegten Knotenanzahl hinzugefügt wurde. Motiviert durch Szenarien in der Praxis wurde die Basisstation in einem der Eckpunkte der Simulationsfläche mit der Koordinate (0,0) positioniert. Diese Positionierung führt aufgrund der multi-hop Kommunikation zu einem erhöhten Kommunikationsaufkommen in der Nähe der Basisstation und stellt daher ein Worst-Case-Szenario hinsichtlich der Skalierbarkeit dar.

Simuliert wurde über eine Dauer von 3000 s und einer zusätzlichen Einschwingphase von 100 s. Diese Phase ermöglicht dem eingesetzten Routing-Protokoll den Aufbau des Routing-Baums. In jeder Simulationsreihe wurden zehn Repli-

kationen durchgeführt. Eine Übersicht über alle oben genannten Simulationsparameter findet sich in der Tabelle 6.8.

Als Metrik zur Bewertung der Skalierbarkeit wurde die Paketankunftsrate (PDR) der Basisstation herangezogen. Dazu wird die Gesamtzahl der während der Simulation von der Basisstation empfangenen Sensordaten-Pakete der Gesamtzahl generierte Sensordaten-Pakete gegenübergestellt, die sich aus der Anzahl der Sensorknoten, der Senderate und der Simulationszeit ergibt, d.h. es gilt:

$$PDR = \frac{\#\text{empf. Sensordaten-Pakete}}{\#\text{Knoten} \cdot \text{Simulationszeit} \cdot \text{Senderate}}.$$

6.6.3 Ergebnisse

Die Skalierbarkeitssimulationen haben gezeigt, dass die Simulation mit großer Knotenanzahl zwar prinzipiell in jeder der drei betrachteten WSN-Simulationsumgebungen möglich ist, jedoch gravierende Unterschiede in der Performance der Simulatoren bestehen. Unter Verwendung des Simulators Cooja in Kombination mit MSPSim, das den für einen TelosB-Knoten mit MSP430 Chipsatz kompilierten Binärcode emuliert, wurde in dem Szenario mit 100 Sensorknoten eine mittlere PDR von lediglich 11,75 ± 2,23% ermittelt. Die Laufzeit der Simulation beträgt dabei auf einem Standardsystem mit einem Intel Core 2 Duo T7700 2,4 GHz Prozessor und 4 GB Arbeitsspeicher ca. 4–5 Stunden. Eine deutlich geringere Performance wird dagegen in dem Szenario mit 500 Sensorknoten erreicht. Hier wurde ein Faktor von ca. 100 ermittelt, um den sich die Echtzeit von der Simulationszeit unterscheidet. Aus diesem Grund erscheint eine Simulation des Contiki-Codes in Cooja für die Szenarien mit 1000 und 2000 Sensorknoten inklusive Replikationen im zeitlich begrenzten Projektrahmen als nicht praktikabel und wurde nicht weiter verfolgt.

Parameter:	**Belegung:**
Anzahl Sensorknoten:	100, 500, 1000 bzw. 2000
Applikation:	periodische Sensordaten-Übertragung
Senderate:	1 Paket/5 s
Kommunikationsmuster:	Convergecast
Routing-Protokoll:	baumbasiert, Anycast
Simulationsfläche:	300 m × 300 m
Kommunikationsreichweite:	50 m
Positionierung:	randomisiert
Position Basisstation:	(0,0)
Simulationsdauer:	3000 s
Einschwingphase:	100 s
Anzahl Replikationen:	10

Tabelle 6.8: Übersicht über die Parameter der vier betrachteten Szenarien zur simulativen Evaluation der Skalierbarkeit.

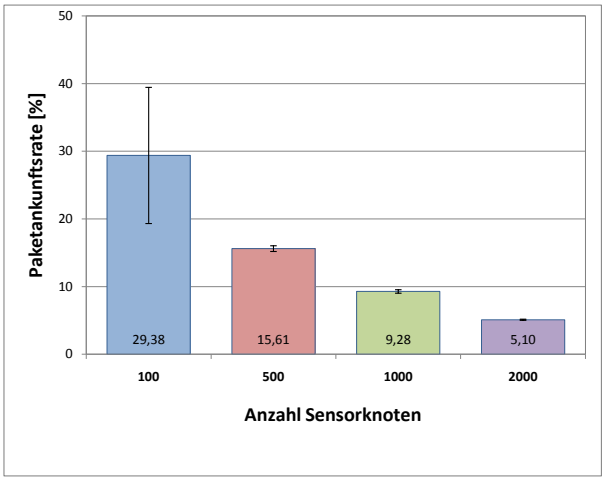


Abbildung 6.19: Die mittlere PDR der Basisstation in Abhängigkeit von der Anzahl in der Shawn-Simulation eingesetzten Sensorknoten.

Auch die Simulation der TinyOS-Applikation in dem WSN-Simulator TOSSIM wird von der geringen Performance des Simulators beeinträchtigt. Die Laufzeit des 100-Knoten-Szenario beträgt auf dem System, das ebenfalls für die Cooja-Simulation verwendeten wurde, ca. 7–8 Stunden. Der sich aus dieser Laufzeit ergebende Faktor, um den sich die Echtzeit von der Simulationszeit unterscheidet, entspricht dem Ergebnis der in [100] durchgeführten Performance-Untersuchung, in der ein zur Knotenanzahl exponentielles Wachstum gezeigt werden konnte. Aufgrund der geringen Performance wurde analog zu den Simulationen in Cooja ebenfalls auf weitere Simulationen mit einer höheren Anzahl von Knoten in TOSSIM verzichtet. Darüber hinaus bietet TOSSIM keine Möglichkeit, Knoten gezielt in der Simulationsfläche zu platzieren. Anstelle der geographischen Position erwartet der Simulator eine Link-Qualitäts-basierte Eingabe der zu simulierenden Topologie. Folglich ist es nicht ohne weiteres möglich, die Basisstation in den Eckpunkt der Simulationsfläche zu platzieren. Daher wurde diese zusammen mit den restlichen Sensorknoten randomisiert in der Fläche positioniert. Die zu untersuchende Skalierbarkeit der Sensorapplikation bzw. die Skalierbarkeit des eingesetzten Routing-Protokolls ist jedoch maßgeblich von der Position der Basisstation abhängig, da eine zentral positionierte Datensenke eine gleichmäßigere verteilte Netzwerklast impliziert als eine Datensenke am Rand der Sensornetzfläche. Dies beeinflusst die Vergleichbarkeit der Simulationsergebnisse und beeinträchtigt massiv die Aussagekraft der Skalierbarkeitsevaluation in TOSSIM, dessen weitere Durchführung nicht sinnvoll erscheint.

Im Gegensatz zu der beobachteten geringen Performance der Simulatoren Cooja und TOSSIM wurde bei dem Wiselib-Simulator Shawn, der sich durch seine effiziente Implementierung auszeichnet, eine bessere Skalierbarkeit der Performance in der Anzahl simulierter Sensorknoten beobachtet. Dies ermöglicht das Durchführen der Simulationen auch mit 2000 Knoten und einer praktikablen Laufzeit. Die Ergebnisse der Shawn-Simulationen sind in Abbildung 6.19 visua-

lisiert, die den signifikanten Abfall der insgesamt niedrigen PDR bei steigender Anzahl von Sensorknoten im Netzwerk verdeutlicht. Während die PDR bei einer Anzahl von 500 Knoten bei etwa 30% liegt, kann mit einer Anzahl von 2000 Knoten eine PDR von lediglich ca. 5% erreicht werden. Die schlechte PDR dieser Anwendung schon bei geringer Anzahl von Sensorknoten (100 Knoten) und die unzureichende Skalierbarkeit der Sensorapplikation sind primär auf die beschränkte Bandbreite des Sensornetzes und auf die damit verbundenen, von der Sensordaten-Übertragung verursachten Überlast im Netzwerk zurückzuführen. Aus der relativ hohen Dichte des simulierten WSN resultiert insbesondere bei 2000 Knoten eine massive Beeinträchtigung durch Interferenzen, die von dem in Shawn verwendeten IEEE 802.15.4 CSMA Transmission-Modell berücksichtigt werden. Zudem zeigt die hohe Varianz der PDR im 100-Knoten-Szenario den starken Einfluss der randomisierten Positionierung der Sensorknoten, mit der bei zu geringer Knotenanzahl die Flächenabdeckung bzw. die Netzwerkkonnektivität nicht gewährleistet werden kann.

Die Simulationsergebnisse lassen vermuten, dass alleine durch eine Reduktion der Netzwerklast, z.B. durch Verringern der Senderate, die durch Aggregation und In-network-processing ermöglicht werden könnte, mit den in der Praxis angestrebten Sensornetzgrößen keine zufriedenstellende Skalierbarkeit erreichbar ist. Einen möglichen Ansatz, die Skalierbarkeit zu verbessern, stellt der Einsatz mehrerer Basisstationen dar, sowie das Gruppieren (engl. Clustering) von Sensorknoten bei Nutzung von Frequency Division Multiple Access (FDMA)-Verfahren.

6.6.4 Simulative Evaluation eines Routing-Angriffs

In Abschnitt 6.2 ist dargelegt, weshalb eine Evaluation des implementierten Sinkhole-Angriffs im realen Netz nicht durchgeführt werden konnte. Ein wesentlicher Aspekt ist die Platzierung der Sensorknoten bzw. dem Erzeugen der gewünschten logischen Topologie. Darüber hinaus führen komplexe Signalausbreitungseffekte in der Praxis häufig zu unvorhersehbaren Link-Qualitäten. Dagegen bieten Simulationen generell eine kontrollierte und berechenbare Umgebung. Daher wurden die obigen Simulationen der iSense-Applikation im Simulator Shawn zur Evaluation des Einflusses eines exemplarischen Routing-Angriffs auf das eingesetzte baumbasierte Routing-Protokoll erweitert. Dazu wurde ein Blackhole-Angreifer, der Routing-Pakete mit optimale Link-Qualitäten fälscht, zentral in der Simulationsfläche platziert. Die zentrale Positionierung des Angreifers sorgt für maximalen Einfluss des ausgehenden Angriffs. Die Simulationen wurden mit den identischen Parametern (siehe Tabelle 6.8) erneut ausgeführt.

Die Ergebnisse der Simulationen bezüglich der PDR der Datensenke und auch des Angreifers sind in Abbildung 6.20 dargestellt. Der höhere Anteil des Angreifers an der Gesamt-PDR, der unabhängig von der Anzahl simulierter Sensorknoten ist, bestätigt den Vorteil der zentralen Positionierung des Angreifers gegenüber der Randposition der Basisstation. Der Vergleich mit den Ergebnis-

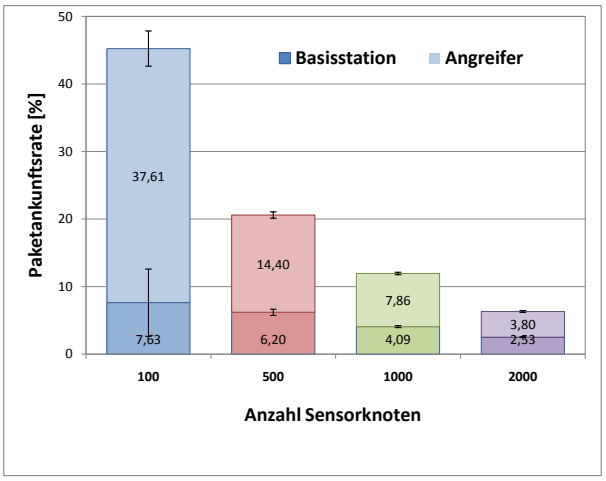


Abbildung 6.20: Die mittlere PDR der Basisstation und des Angreifers in Abhängigkeit von der Anzahl in der Shawn-Simulation eingesetzten Sensorknoten.

sen der Simulationen aus dem vorherigen Abschnitt (Abbildung 6.19), in denen kein Angriff auf die Sensorapplikation durchgeführt wurde, verdeutlicht anhand der signifikanten Reduktion der PDR die Beeinträchtigung der Zuverlässigkeit der Anwendung durch den Routing-Angriff und damit die Relevanz einer geeigneten Gegenmaßnahme.

6.7 Egoistische Sensorknoten

In diesem Abschnitt wird darauf eingegangen, was egoistische Sensorknoten sind, wie sie in einer Sicherheitsarchitektur einzuordnen sind und welche Gegenmaßnahmen ergriffen werden können.

In einem dezentralen Netzwerk, wie einem WSN, hängt die Lebensdauer des gesamten Netzes maßgeblich von der Lebensdauer der einzelnen Knoten ab. Fällt ein Knoten aus, steht dieser nicht mehr für die Weiterleitung von Sensordaten zur Senke bereit. Fallen genügend viele Knoten aus, wird das gesamte Netzwerk unbrauchbar, obwohl einzelne Knoten eventuell noch ausreichende Ressourcen hätten. Ziel ist es also in einem WSN die Ressourcen aller Knoten in gleichen Maßen zu verbrauchen und somit die Lebensdauer des gesamten Netzes zu maximieren. Hierzu ist es notwendig, dass alle Knoten miteinander kooperieren. Eine solche Kooperation kann sich bspw. darin äußern, dass das Routing einvernehmlich angepasst wird, um Knoten zu entlasten, die bisher stärker durch das Weiterleiten von Paketen beansprucht wurden (vgl. Abschnitt 2.2.2).

Im Gegensatz zu kooperierenden Knoten versuchen egoistische Knoten ihr Verhalten allein mit Hinblick auf ihre eigenen Ressourcen zu optimieren. Im Allgemeinen wird dies durch Missachten von Regeln oder Senden von falschen bzw. keinen Paketen erreicht [45, 88]. Im Extremfall mündet ein solches Verhalten in

einen Angriff, wie z.B. dem Rushing (siehe Abschnitt 5.4.1) oder dem Sinkhole-Angriff (siehe Abschnitt 5.5.1). Bei einem egoistischen Knoten handelt es sich also nicht um einen speziellen Angriff der eine spezielle Gegenmaßnahme erfordert, sondern vielmehr um eine Verhaltensweise, der in Ansätzen mehrere der vorgestellten Angriffsmethoden zugrunde liegen können.

Im Gegensatz zu allgemeinen drahtlosen Netzen gilt in WSNs in der Regel jedoch die Annahme, dass alle Knoten einer einzigen Instanz unterliegen und somit eine Kooperation der Knoten außer Frage steht. Daher kann hier davon ausgegangen werden, dass es sich bei egoistischen Knoten stets um einen Angreifer handeln muss (siehe auch Abschnitt 5.6.3). Folglich kann egoistisches Verhalten eines Knotens als Angriff gedeutet und entsprechende Maßnahmen ergriffen werden.

Trifft die Annahme, dass alle Knoten eines WSNs einer einzigen Instanz unterliegen, nicht zu, sollte eine sinnvolle Unterscheidung zwischen egoistischen Knoten und Angreifern getroffen werden. Da der Übergang jedoch fließend ist, muss eine entsprechende Behandlung ebenfalls graduell erfolgen. Hierfür existiert der Ansatz, der in [60] beschrieben wird: zunächst wird stets kooperiert und nur auf egoistisches Verhalten eines Sensorknotens hin die Kooperation mit diesem Kommunikationspartner sukzessive eingestellt. Bei Wiederbeginn der Kooperation von Seiten des (egoistischen) Kommunikationspartners, wird das Einstellen der Kooperation wieder aufgehoben.

6.8 Zusammenfassung und Fazit

Zunächst wurde in diesem Kapitel das Intrusion Detection System als zentrale Komponente der Sicherheitsarchitektur vorgestellt. Dieses verarbeitet die Alarmnachrichten der einzelnen Angriffsdetektoren auf den Sensorknoten im Netz. Durch den modularen Aufbau des IDS lassen sich Module für weitere Detektoren auf einfache Weise ergänzen. Auf die Beschreibung dieser Kernkomponente der Sicherheitsarchitektur folgte die Vorstellung der Realisierung und Evaluation ausgewählter Angriffe und Gegenmaßnahmen.

Basierend auf einer detaillierten Codeanalyse wurde der Sinkhole-Angriff implementiert. Dies erwies sich als eine unerwartete Herausforderung, da eine klassische Realisierung des Angriffs nur geringen Erfolg versprach. Daher wurde mittels einer Adaptierung des Ansatzes versucht, die Erfolgschancen für einen Angriff zu erhöhen. Letztlich zeigte sich aber, dass, aufgrund der Robustheit bzw. Trägheit des Protokolls kombiniert mit Signalausbreitungseffekten, der Einfluss der Angriffsansätze nicht zuverlässig reproduzierbar war. Daher war ebenso eine Implementierung der geplanten Gegenmaßnahme nicht sinnvoll.

Um Jamming-Angriffe zu erkennen, wurde ein Carrier-Sense-Time-Sensor realisiert. Die Detektionsfähigkeit des Sensors wurde anschließend im Testbed evaluiert. Weiterhin wurden verschiedene symmetrische und asymmetrische kryptographische Verfahren diskutiert und implementiert. Die Evaluation dieser Ver-

fahren hat zum einen (wie erwartet) gezeigt, dass eine Implementierung in Hardware deutliche Geschwindigkeitsvorteile bringt. Zum anderen wurde deutlich, dass der Geschwindigkeitseinbruch bei der Benutzung von asymmetrischer Verschlüsselung (inkl. ECC) zwar recht deutlich für einige Anwendungen ist, aber dennoch in einem akzeptablen Rahmen liegt. Dieser könnte voraussichtlich noch weiter verringert werden, wenn die Verfahren bereits in Hardware implementiert sind.

Basierend auf der Diskussion und Evaluation der kryptographischen Verfahren wurde das Secure OTAP entwickelt, welches asymmetrische Kryptographie als Basis für signatur-gestütztes OTAP verwendet. Die experimentelle Evaluation zeigte, dass zum einen die Zeit zur Signaturverifikation in Relation zur Gesamtdauer der Übertragung eines kompletten Images klein genug ist. Zum anderen ist der Overhead vernachlässigbar, der durch die Signatur in den Paketen erzeugt wird.

Im Anschluss an die in der umgesetzten Testumgebung durchgeführten Evaluationen wurden simulative Untersuchungen angestellt. Zunächst wurde die Skalierbarkeit der exemplarisch realisierten Sensorapplikation (in der Größe des Sensornetzes) durch Simulationen mit bis zu 2000 Sensorknoten evaluiert. Dabei wurde erkannt, dass schon bei einer Anzahl von lediglich 100 Sensorknoten eine massive Überlast im WSN zu erwarten ist und dass es weitere Maßnahmen bedarf, der Überlastung des Netzwerks entgegenzuwirken. Anschließend wurde der Einfluss eines Routing-Angriffs, dessen Evaluation im realen Netz nicht durchführbar war, simulativ in den zuvor betrachteten Szenarien evaluiert und die signifikante Beeinträchtigung der Funktionalität des Sensornetzes durch diesen Angriff verdeutlicht.

Abschließend wurden Schutzmaßnahmen gegen unkooperative und egoistische Sensorknoten beschrieben und in den Kontext der erfolgten Arbeiten eingeordnet.

Im Rahmen weiterführender Arbeiten wäre es sinnvoll, das vorhandene Secure OTAP zu verbessern und auszubauen. Ferner wäre es wünschenswert, den Trusted Sensor Node (TSN) in die Labortestumgebung zu integrieren und dort zu evaluieren. Einfache Routingangriffe, wie z.B. das Root Node Spoofing, die Erfolg versprechend sind, könnten so erkannt bzw. vermieden werden.

Teil III

Lokalisierungs- und Vitaldatenanwendung

7 Überblick über Lokalisierungsverfahren

Dieses Kapitel liefert einen Überblick über Lokalisierungsverfahren mit besonderem Augenmerk auf ihre Eignung für drahtlose Sensornetze.

In der wissenschaftlichen Literatur wurde in den vergangenen Jahren eine Vielzahl von Verfahren zur Lokalisierung vorgestellt. Sie sind erforderlich, da die Sensorknoten nach ihrer Ausbringung im Feld im Allgemeinen keine Informationen über ihre räumliche Lage haben. Lokalisierungsverfahren dienen dazu, eben solche Positionsinformationen zu ermitteln.

Zahlreiche Arbeiten betonen die Relevanz von Positionsinformationen für die Funktion von Sensornetzanwendungen. Gerade in Sensornetzen sind ihre Verwendungsmöglichkeiten vielfältig. Zum einen ist für die Bewertung von Messdaten der Ort ihres Entstehens inhärent relevant, zum anderen können mit Hilfe von Ortsinformationen räumlich benachbarte Daten zusammengefasst werden [104]. Spezielle Aufgaben wie das Verfolgen von Objekten benötigen ebenfalls Positionsdaten, um die Bewegungsrichtung ermitteln zu können [165].

Darüber hinaus können Positionsinformationen zum sogenannten *geographischen Routing* genutzt werden, das heißt um Daten auf Basis der Position der Zielknoten weiterzuleiten [90, 95].

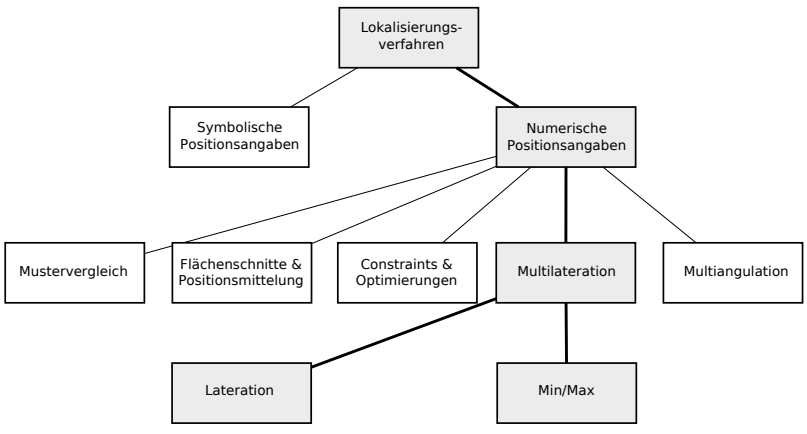


Abbildung 7.1: Überblick über die Lokalisierungsverfahren.

Abbildung 7.1 gibt einen Überblick über die verschiedenen Verfahren zur Erlangung von Positionsbewusstsein, die im folgenden Abschnitt vorgestellt werden. Dabei sind die besonders interessanten Verfahren grau hinterlegt. Verfahren auf

Basis der Multilateration scheinen nach der folgenden Analyse für Sensornetze insbesondere geeignet.

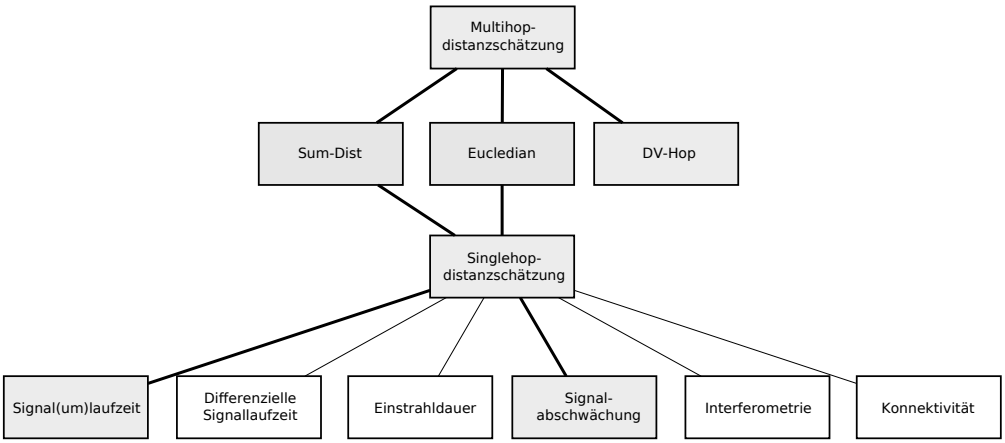


Abbildung 7.2: Überblick über die Distanzschätzverfahren.

Der zweite Abschnitt dieses Kapitels diskutiert Distanzschätzverfahren, da diese für die Multilateration von entscheidender Wichtigkeit sind. Abbildung 7.2 gibt einen Überblick über die verschiedenen Distanzschätzverfahren, wiederum sind die besonders interessanten Verfahren grau hinterlegt.

7.1 Lokalisierungsverfahren

Im Hinblick auf die Art der Positionsangaben lassen sich numerische von symbolischen Angaben unterscheiden.

Erstere drücken die Positionen in Form von Koordinaten aus. Das hat den Vorteil, dass die entsprechenden Angaben sehr abstrakt sind und sich somit für nahezu jede Anwendung eignen. Darüber hinaus lassen sich aus den Koordinaten weitere Angaben wie beispielsweise Distanzen direkt ableiten. Diese Eigenschaften haben dazu geführt, dass die überwältigende Mehrheit der Verfahren zur Positionsbestimmung numerische Angaben liefert. Abschnitt 7.1.1 stellt deren unterschiedliche Funktionsweisen vor.

Symbolische Positionsangaben hingegen schließen bereits eine gewisse Interpretation ein, sie lauten etwa „am Rand des Sensornetzes“ oder „Raum X in Gebäude Y“. Beispiele für solche Verfahren sind [57], das die Randlage von Sensorknoten ermittelt, oder [92], das als Ergebnis die Lage der Sensorknoten in Straßen- und Kreuzungsbereichen eines Straßennetzes ermittelt und benennt. Die entsprechenden Ansätze haben den Vorteil, dass sie auf den Einsatz von Heuristiken und meist fehlerbehafteten Distanzschätzungen verzichten können, die bei numerischen Verfahren speziell in schwierigen Szenarien im Extremfall zum kompletten Versagen führen können [30]. Andererseits sind die resultierenden symbolischen Positionsangaben nicht für alle Anwendungsbereiche geeignet

und konnten sich deshalb nicht auf breiter Front durchsetzen. Aus diesem Grund werden die symbolischen Verfahren im Weiteren nicht näher betrachtet.

7.1.1 Verfahren zur Errichtung von numerischem Positionsbewusstsein

Nahezu alle Verfahren, die dazu dienen, die Knoten eines Sensornetzes mit Informationen über ihre Position in Form von Koordinaten zu versorgen, gehen davon aus, dass einige Sensorknoten ihre Position bereits kennen. Diese a priori positionsbewussten Sensorknoten werden im Weiteren *Ankerknoten* genannt.

Die Positionsinformationen der Anker können aus unterschiedlichsten Quellen stammen. Die Anker können beispielsweise gezielt an einer bekannten Position ausgebracht werden und die entsprechenden Daten manuell mitgeteilt bekommen. Dieses Vorgehen steigert jedoch den Aufwand bei der Ausbringung des Sensornetzes erheblich.

Alternativ können die Anker mit einer separaten Technik zur Positionsbestimmung wie beispielsweise dem GPS [75] ausgestattet sein. Nachteile dieser Variante sind, dass die entsprechende zusätzliche Hardware sich negativ auf Baugröße, Energieverbrauch und Preis auswirkt. Somit ist es vorteilhaft, den Anteil der Ankerknoten relativ zur Gesamtanzahl der Sensorknoten möglichst gering zu halten.

Auf den Ankern aufbauend ist es Inhalt der meisten in diesem Abschnitt vorgestellten Verfahren, die Informationen über die Positionen der Anker gemeinsam mit zusätzlichen, meist relativen räumlichen Information so an alle anderen Sensorknoten zu verteilen, dass diese ihre eigene Position mindestens näherungsweise bestimmen können.

Dieses kann innerhalb des Sensornetzes, also verteilt auf den einzelnen Sensorknoten, geschehen oder aber auf einer zentralen Berechnungskomponente. Dem Vorteil der zentralisierten Variante, dass global auf alle verfügbaren Informationen zurückgegriffen werden kann, stehen die Nachteile eines erhöhten Kommunikationsbedarfs und schlechtere Skalierbarkeit gegenüber.

Im Folgenden werden unterschiedliche Gruppen von Verfahren zur Errichtung von Positionsbewusstsein vorgestellt und in ihrer Funktionsweise beschrieben.

Mustervergleich

Verfahren zur Positionsbestimmung auf Basis von Mustervergleichen, in der Literatur vielfach auch *Fingerprinting* genannt, basieren auf der Annahme, dass für den Bereich, innerhalb dessen Positionen bestimmt werden sollen, eine Datenbasis existiert. Sie verzeichnet die messbaren Eigenschaften jeder Position innerhalb dieses Bereiches.

Ein Gerät, das seine Position bestimmen will, misst zunächst die entsprechenden Eigenschaften am momentanen Aufenthaltsort. Dann sucht es innerhalb der Datenbasis diejenigen Koordinaten, deren gespeicherte Eigenschaften am besten zu den gemessenen passen und nimmt diese Koordinaten als aktuellen Aufenthaltsort an.

In der Literatur finden sich hauptsächlich Beiträge, die Signalcharakteristika von WLAN-Zugangspunkten zur Positionsbestimmung nutzen (zum Beispiel [158]). Andere Publikationen beschreiben Verfahren, die das Bild einer auf einem mobilen Roboter montierten Kamera zur Positionsbestimmung mit bekannten Bildern vergleichen [58].

Der zentrale Nachteil dieser Verfahren liegt im immensen initialen Aufwand, der für die Erstellung der verwendeten Datenbasis erforderlich ist. So muss der gesamte Bereich im Vorfeld vermessen und die Messergebnisse in der Datenbasis aufgezeichnet werden. Daher lässt sich die für Sensornetze gewünschte spontane Einsatzbereitschaft mit diesem Verfahren nicht erzielen. Darüber hinaus nimmt die Datenbasis große Mengen Speicherplatz in Anspruch, so dass davon auszugehen ist, dass sie nicht auf jedem Gerät zur Verfügung stehen kann, sondern nur auf einer zentralen Komponente vorgehalten werden kann. Die Sensorknoten müssten somit die Messdaten an diese Instanz weiterleiten, und die zentral berechneten Positionsdaten müssten an die Sensorknoten zurückgesandt werden.

Nicht nur der hohe initiale Aufwand, auch das hohe Kommunikationsaufkommen und der resultierende Energieverbrauch sowie die mangelnde Skalierbarkeit einer solchen zentralisierten Lösung führen dazu, dass dieses Verfahren für Sensornetze nicht geeignet ist.

Flächenschnitte und Positionsmittelung

Eines der einfachsten Verfahren, um Sensorknoten mit Positionsinformationen zu versorgen, ist die Positionsmittelung. Dabei wird davon ausgegangen, dass jeder Sensorknoten mehrere Anker zu seinen direkten Nachbarn zählt.

Jeder Anker sendet seine Position mittels eines Broadcasts an seine Nachbarn, die ihrerseits alle empfangenen Positionen speichern. Aus diesen Positionsinformationen bestimmen sie dann ihre eigene Position.

Dies geschieht im einfachsten Fall ausschließlich durch eine Mittelwertbildung, das heißt die Position P_i eines Sensorknotens i ergibt sich aus den empfangenen Positionen $B_j = (x_j, y_j, z_j)$ durch

$$P_i = (x_i, y_i, z_i) = \frac{1}{n} \sum_{j=1}^{n} B_j. \tag{7.1}$$

Dieses *Centroid Location* genannte Verfahren wird in [28] beschrieben und kommt ebenfalls in [27, 91] zum Einsatz. Diese Art der Positionsberechnung ist

mathematisch besonders anspruchslos und lässt sich auch auf leistungsschwachen Mikrokontrollern leicht umsetzen. Allerdings hat sie den Nachteil, dass alle Sensorknoten, die die gleichen Anker zu ihren Nachbarn zählen, die exakt selbe Position berechnen, auch wenn sie sich in Wirklichkeit an unterschiedlichen Orten befinden.

Um diesen Nachteil zu kompensieren, wurden Erweiterungen vorgeschlagen, die die Positionen der Anker unterschiedlich gewichten [21]. Bei diesem *Weighted Centroid* oder *gewichtete Mittelung* genannten Ansatz ergibt sich die Position P des Sensorknotens i durch multiplikative Gewichte w_{ij} bezüglich der Ankerpositionen B_j gemäß der Formel

$$P_i = (x_i, y_i, z_i) = \frac{\sum_{j=1}^{n} (w_{ij} \cdot B_j)}{\sum_{j=1}^{n} w_{ij}}. \tag{7.2}$$

Das Gewicht w_{ij} ergibt sich aus der Distanz d_{ij} zwischen Ankern j und Sensorknoten i, dabei kann der Einfluss der Distanz mit Hilfe eines Exponenten e angepasst werden:

$$w_{ij} = \frac{1}{(d_{ij})^e}. \tag{7.3}$$

Konkret ließe sich beispielsweise der Signalstärkewert (auch *Link Quality Indicator* (LQI)) der Anker als Gewicht einsetzen, er ergibt sich $w_{ij} = LQI_{ij}$. In [17] wird dieses Verfahren verfeinert, um allgemeine Schwankungen in der Stärke der Signale von den Ankern auszugleichen. Die Autoren schlagen vor, einen gewissen Bruchteil q der niedrigsten Signalstärke von den Signalstärken aller Anker abzuziehen; die Position ergibt sich damit zu

$$P_i = (x_i, y_i, z_i) = \frac{\sum_{j=1}^{n} ((LQI_{ij} - q \cdot min(LQI_{i,1..n})) \cdot B_j)}{\sum_{j=1}^{n} (LQI_{ij} - q \cdot min(LQI_{i,1..n}))}. \tag{7.4}$$

Der in [153] vorgestellte Ansatz arbeitet ähnlich, verwendet jedoch anstelle der gewichteten Mittelung die sogenannte *Monte-Carlo-Integration*.

Dem Vorteil der geringen Berechnungskomplexität steht bei all diesen Verfahren der Nachteil gegenüber, dass jeder Sensorknoten mehrere Anker in der direkten Nachbarschaft benötigt. Darüber hinaus können naturgemäß nur solche Positionen Ergebnis der Mittelung sein, die innerhalb der konvexen Hülle der Ankerpositionen liegen, obwohl dieses nicht zwangsläufig der tatsächlichen Lage entsprechen muss.

Constraints und Optimierungen

Eine weitere Gruppe von Verfahren basiert auf der Auswertung von Beschränkungen der räumlichen Lage von Sensorknoten untereinander sowie zwischen

Sensorknoten und Ankern. Im Rahmen eines globalen Optimierungsprozesses können dann die Positionen der Sensorknoten näherungsweise bestimmt werden.

Das in [78] vorgestellte, zentralisierte Verfahren geht davon aus, dass die Anker ihre Positionen broadcasten. Empfangen mehrere Sensorknoten diese Informationen, lässt sich zwischen ihnen ein *Constraint* aufstellen, das heißt, ihre möglichen relativen Positionen zueinander werden eingeschränkt. Aus allen diesen Constraints lassen sich im Rahmen einer globalen Optimierung die Positionen der Sensorknoten berechnen.

Ähnlich arbeitet der Ansatz aus [44]. Hier werden die Nachbarschaftsbeziehungen aller Sensorknoten in einem sogenannten *Konnektivitätsgraphen* dargestellt. Mit Hilfe eines zentralen Optimierungsschrittes, der auf dem Verfahren der *linearen Programmierung* basiert, werden dann die resultierenden Knotenpositionen derart bestimmt, dass sie die Bedingungen des Konnektivitätsgraphen erfüllen.

Eine Erweiterung dieser beiden Ansätze wird in [156] beschrieben. Zusätzlich zu den Konnektivitätsbeziehungen wird hier berücksichtigt, dass Sensorknoten, die nicht direkt miteinander kommunizieren können, einen gewissen Mindestabstand haben müssen. Nach Aussagen der Autoren reduziert die Hinzunahme dieser Informationen den Fehler der berechneten Positionen um etwa ein Drittel.

Während die obigen Ansätze auf der binären Information der Konnektivität beruhen, werten die im Folgenden beschriebenen Verfahren Distanzen zwischen den Sensorknoten aus.

In [151] wird ein *MDS-MAP* genanntes, zentralisiertes Verfahren vorgeschlagen, das auf dem Prinzip der *Multidimensionale Skalierung* (MDS) [23] beruht. Es berechnet die relativen Positionen der Sensorknoten untereinander mit einer Komplexität von $O(n^3)$ unter der Annahme, dass alle relativen Distanzen zwischen den Sensorknoten bekannt sind. In einem weiteren Berechnungsschritt der Komplexität $O(n)$ können die relativen Knotenpositionen mit Hilfe der Ankerpositionen dann in absolute Positionen überführt werden.

Mehrere Autoren schlagen vor, Positionsbestimmung auf Basis von Distanzinformationen mit Hilfe der *Maximum-Likelihood-Methode* durchzuführen. Während in [125] ein Verfahren zur Positionsbestimmung in mittels WLAN verbundenen P2P-Netzen vorgestellt wird, beschreibt [123] die Adaption für Sensornetze. Ausgehend von einigen wenigen bekannten Ankerpositionen sowie Informationen über die Distanzen zwischen benachbarten Sensorknoten und Ankern werden die Positionen zentral berechnet. Ähnliche Ansätze werden in [132, 133] beschrieben.

In [124] wird untersucht, inwieweit global optimierende Verfahren, die aus empfangenen Signalstärken erlangte Distanzinformationen verwenden, bessere Ergebnisse erzielen als solche Verfahren, die lediglich auf Konnektivitätsinformationen beruhen. Die Autoren kommen zu dem Ergebnis, dass erstere um bis zu

50% geringere Fehler erzielen können als letztere, was den Wert von Distanzinformationen für die Positionsbestimmung unterstreicht.

Allen bisher besprochenen Verfahren ist gemein, dass sie auf zentralen Berechnungen beruhen. Dieses ist meist schon aufgrund der immensen Komplexität der eingesetzten Verfahren erforderlich, die die Rechen- und Speicherkapazitäten eines Sensorknoten weit überschreiten. Das bringt jedoch mit sich, dass alle im Netz ermittelten Informationen über Konnektivität oder Distanzen zur Berechnungsinstanz geleitet werden müssen. Die berechneten Positionen müssen im Anschluss wiederum den einzelnen Sensorknoten mitgeteilt werden, so dass ein erheblicher Kommunikationsaufwand entsteht.

Dieser Nachteil führt dazu, dass die meisten Verfahren nur bedingt für Sensornetzwerke geeignet sind. Dieses schlägt sich auch darin nieder, dass praktisch keine entsprechenden Implementierungen existieren.

Es gibt jedoch auch verteilt arbeitende Verfahren, die auf Einschränkungen möglicher Positionen oder Optimierungen basieren.

Das in [175] beschriebene *Calamari-System* verwendet neben anderen Verfahrensbestandteilen, die sich in den Bereich der Multilateration einordnen lassen, einen *Resolution of Forces* genannten Ansatz, um bereits berechnete, grobe Positionsinformationen iterativ zu verfeinern. Hierbei wird das Spannungsverhältnis zwischen Distanzen zu Ankern einerseits und der im Vorfeld bestimmten Position andererseits als eine Kraft mit Richtung und Betrag aufgefasst, die den Sensorknoten bildlich betrachtet zu seiner korrekten Position hinzieht. Durch iterative Anpassung der berechneten Positionen wird nun versucht, alle Kräfte im Sensornetz möglichst auf Null zu reduzieren.

Obwohl das in [71] beschriebene Verfahren völlig anders arbeitet als die bisher besprochenen, lässt es sich doch am ehesten in diesen Abschnitt einordnen, da es die mögliche Position eines Sensorknotens sukzessive einschränkt. Die Autoren gehen davon aus, dass die Anker im Netz über eine besonders hohe Kommunikationsreichweite verfügen, so dass jeder Sensorknoten seine Distanz zu vielen Ankern sowie deren Positionen kennt. Die Sensorknoten bilden nun aus den Positionen von jeweils 3 Ankern Dreiecke und prüfen, ob sie innerhalb dieser liegen. Dieses geschieht mit Hilfe des sogenannten *Approximate Point-In-Triangulation Test (APIT)*. Der besagt, dass ein Sensorknoten innerhalb des Dreiecks aus Ankern liegt, wenn keiner seiner Nachbarn näher an allen drei Ankern liegt als er selber.

Abbildung 7.3 verdeutlicht dieses an einem Beispiel. Bezüglich des Dreiecks aus den Ankern 1, 2 und 3 liegt keiner der grauen Nachbarn näher an allen drei Ankern als der weiße Sensorknoten, bezüglich der Anker 2, 3 und 4 jedoch liegt D näher an allen drei Ankern. Somit kann der weiße Sensorknoten schlussfolgern, dass er innerhalb des ersten, aber außerhalb des zweiten Dreiecks liegt. Alle Sensorknoten bilden nun den Schnitt all jener Dreiecke, innerhalb derer sie liegen, und schränken so ihre Position sukzessive ein. Schließlich platzieren sie sich in die Mitte der verbleibenden Fläche.

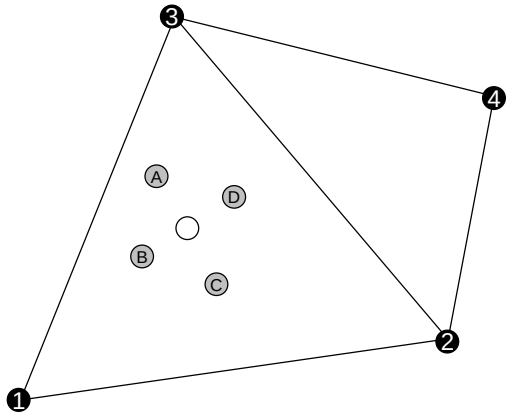


Abbildung 7.3: Der Approximate Point-In-Triangulation Test.

Multilateration

Das meistverbreitete Verfahren zur Positionsbestimmung ist die Multilateration. Hierbei wird die gesuchte Position der nicht positionsbewussten Sensorknoten $p = (x, y, z)$ aus den bekannten Positionen $p_i = (x_i, y_i, z_i)$ der Anker sowie den Distanzen zu den Ankern d_i berechnet.

Lokationsverfahren, die nach dem Prinzip der Multilateration arbeiten, verlaufen i.A. in drei Phasen [97]:

1. Zuerst werden die Distanzen zwischen den Ankerknoten und den nicht positionsbewussten Sensorknoten ermittelt. Dieses geschieht meist, indem zunächst die Distanzen zwischen benachbarten Sensorknoten bestimmt werden, um diese im Anschluss zu den Distanzen zu den Ankern zusammenzuführen. Entsprechende Verfahren werden in den Abschnitten 7.2.2 bis 7.2.3 vorgestellt.
2. Aus diesen Distanzen und den Ankerpositionen wird dann die Position der Sensorknoten bestimmt.
3. Bei einigen Verfahren schließt sich eine Verfeinerung an. Hier werden große Unterschiede der im vorangegangenen Schritt erlangten Positionsinformationen bei benachbarten Sensorknoten ausgeglichen. Dieses kann zum Beispiel durch eine erneute, lokale Lateration oder Positionsmittelung geschehen.

Typisch für die Positionsbestimmung in Schritt zwei ist die Lateration. Dazu kommen die folgenden Gleichungen zum Einsatz, wobei die gesuchte Position durch die Unbekannten x, y und z beschrieben wird.

$$\begin{aligned} (x_1 - x)^2 + (y_1 - y)^2 + (z_1 - z)^2 &= d_1^2 \\ &\vdots \\ (x_n - x)^2 + (y_n - y)^2 + (z_n - z)^2 &= d_n^2 \end{aligned} \tag{7.5}$$

Dieses Gleichungssystem kann linearisiert werden, indem man die n-te Gleichung von den ersten n-1 Gleichungen subtrahiert.

$$x_1^2 - x_n^2 - 2(x_1 - x_n)x + y_1^2 - y_n^2 - 2(y_1 - y_n)y + z_1^2 - z_n^2 - 2(z_1 - z_n)z = d_1^2 - d_n^2$$

$$\vdots \tag{7.6}$$

$$x_{n-1}^2 - x_n^2 - 2(x_{n-1} - x_n)x + y_{n-1}^2 - y_n^2 - 2(y_{n-1} - y_n)y + z_{n-1}^2 - z_n^2 - 2(z_{n-1} - z_n)z = d_{n-1}^2 - d_n^2$$

Eine Neuordnung der Gleichungen führt zu einem linearen Gleichungssystem der Form $Ap = b$ mit

$$A = \begin{bmatrix} 2(x_1 - x_n) & 2(y_1 - y_n) & 2(z_1 - z_n) \\ \vdots & \vdots & \vdots \\ 2(x_{n-1} - x_n) & 2(y_{n-1} - y_n) & 2(z_{n-1} - z_n) \end{bmatrix}, \tag{7.7}$$

$$b = \begin{bmatrix} x_1^2 - x_n^2 + y_1^2 - y_n^2 + z_1^2 - z_n^2 + d_n^2 - d_1^2 \\ \vdots \\ x_{n-1}^2 - x_n^2 + y_{n-1}^2 - y_n^2 + z_{n-1}^2 - z_n^2 + d_n^2 - d_{n-1}^2 \end{bmatrix}. \tag{7.8}$$

Dieses Gleichungssystem kann nun mittels des Verfahrens der kleinsten Quadrate gelöst werden:

$$p = (A^T A)^{-1} A^T b. \tag{7.9}$$

In Einzelfällen kann es vorkommen, dass die inverse Matrix aufgrund von fehlerbehafteten Distanzen nicht berechnet werden kann, und die Lateration scheitert. In der Mehrheit der Fälle jedoch ist die Lateration erfolgreich und liefert eine Position p. Einen weiteren Plausibilitätstest ermöglicht die Berechnung des sogenannten Residuums

$$r = \frac{\sum_{i=1}^{n} \sqrt{(x_i - x)^2 + (y_i - y)^2 + (z_i - z)^2 - d_i}}{n}. \tag{7.10}$$

Ein großes Residuum signalisiert ein inkonsistentes Gleichungssystem. Daher ist es sinnvoll, die auf diese Weise berechnete Position p zu verwerfen, wenn der Wert des Residuums die mittlere Kommunikationsreichweite überschreitet [97]. Stehen mehr Gleichungen zur Verfügung als zur Positionsberechnung notwendig sind, kann diejenige Menge von Gleichungen zur Berechnung ausgewählt werden, die zum kleinsten Residuum führt.

Die große Anzahl von Publikationen, die Lokationsverfahren nach dem Prinzip der Lateration beschreiben, belegt die hohe Akzeptanz dieses Verfahrens. Beispiele für die Funktion nach dem oben beschriebenen Phasenschema finden sich

in [113,114,117,142,143,146], wobei sich die entsprechenden Verfahren lediglich in Details wie der Art der Distanzschätzung, der Bewertung der Residuen oder der Art der Verfeinerung unterscheiden.

Andere Beiträge verzichten auf das Zusammenführen von Distanzen über mehrere Hops hinweg. Hier ist eine Positionsbestimmung für solche Sensorknoten möglich, die über mindestens drei Anker als direkte Nachbarn verfügen [19].

Um diesen Nachteil zumindest abzuschwächen, wurden verschiedene iterative Ansätze vorgeschlagen [1, 5, 36, 138, 144]. Hier übernehmen Sensorknoten, die bereits eine Position berechnet haben, im Weiteren die Rollen von Ankern, so dass weitere Knoten ihre Position bestimmen können.

Neben der Tatsache, dass die Laterationsberechnung scheitern kann, ist der hohe mathematische Aufwand gerade für leistungsschwache Sensorknoten ein Nachteil. Als Ausweg wurde das Min-Max-Verfahren vorgeschlagen [145, 152].

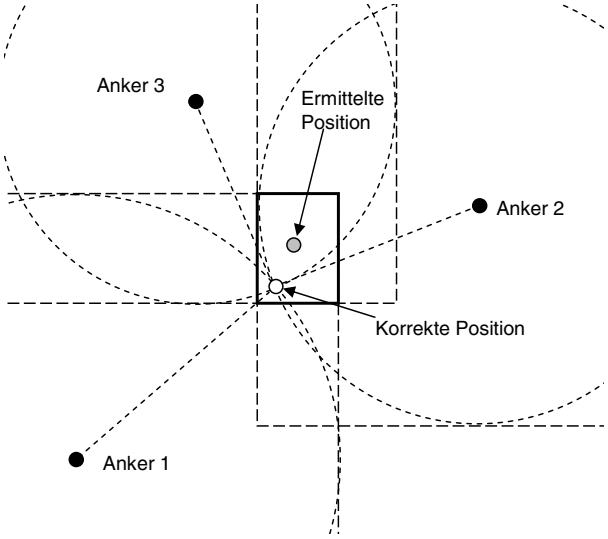


Abbildung 7.4: Positionsbestimmung mit dem Min-Max-Verfahren.

Auch hier wird die Position der Sensorknoten aus Distanzen zu Ankern sowie deren Positionen berechnet (vgl. Abbildung 7.4). Dazu wird um die Ankerpositionen (schwarz ausgefüllte Kreise) jeweils ein Quadrat mit der doppelten Kantenlänge des Abstandes zu diesem Anker gelegt. Die Schnitte dieser Quadrate um alle bekannten Anker ergeben ein Rechteck, in dessen Mitte sich der Sensorknoten positioniert. Gegenüber der klassischen Lateration ist der Berechnungsaufwand erheblich geringer, allerdings wird in aller Regel nicht die exakt richtige Position (weißer Kreis) berechnet, sondern lediglich eine Näherungslösung (grau ausgefüllter Kreis).

Während [152] nur dann eine Position berechnen kann, wenn mindestens drei Anker zu den direkten Nachbarn eines Sensorknoten gehören, ermittelt das *N-hop Multilateration* genannte Verfahren aus [145] auch dann Positionen, wenn die Anker mehrere Hops weit entfernt sind. Dazu verwendet es das *Sum-Dist-*

Schema (siehe auch Abschnitt 7.2.3), das Distanzen über mehrere Hops hinweg addieren kann.

Neben den bisher beschriebenen Verfahren, die einerseits verteilt und andererseits netzweit arbeiten, gibt es auch Vorschläge, das Sensornetz in Cluster zu unterteilen. Innerhalb dieser Cluster werden im ersten Schritt Positionen berechnet, die meist relativ zu einem Cluster-head bestimmt werden. Hierzu kommen prinzipiell die bisher beschriebenen Verfahren zu Einsatz, zum Teil jedoch in abgewandelter Form und ergänzt durch trigonometrische Erwägungen. In einem zweiten Schritt werden dann die bezüglich lokaler Koordinatensysteme angegebenen Positionen in ein globales System überführt. Beispiele für diese Vorgehensweise finden sich etwa in [83] und [142].

Ein problematischer Punkt an der Multilateration ist die Schwierigkeit, die Distanzen zwischen den Knoten und den Ankern zu bestimmen. Dazu kommen *Distanzschätzverfahren* zum Einsatz, deren Genauigkeit die Präzision der ermittelten Positionen maßgeblich beeinflusst. Diese Verfahren arbeiten jedoch häufig ungenau oder langsam, benötigen zusätzliche Hardware wie spezialisierte Sende- und Empfangseinrichtungen oder weisen eine hohe Berechnungskomplexität auf. Die einzelnen Verfahren sowie ihre Vor- und Nachteile werden im Abschnitt 7.2.2 vorgestellt.

Multiangulation

Eine weitere Gruppe bilden die Verfahren nach dem Prinzip der Multiangulation. Hier werden neben der bekannten Lage der Anker Winkelinformationen zur Positionsbestimmung eingesetzt. Die entsprechenden Verfahren wurden meist ursprünglich zur Positionsbestimmung im Bereich der Luft- und Seefahrt entwickelt.

Einen guten Überblick über die möglichen Varianten der Multiangulation gibt [118]. Grundsätzlich lassen sich zwei Methoden unterscheiden:

- *VHF Omnidirectional Radio Range (VOR)* oder auch *Radial*: die Kurzform VOR steht für *VHF Omnidirectional Radio Range*, VHF wiederum heißt *Very High Frequency* - zu Deutsch Ultrakurzwelle (UKW). VOR bedeutet also übersetzt „UKW-Drehfunkfeuer“.
- *Angle of Arrival (AoA)* oder auch *Bearing*

Die VOR-Methode verlangt, dass die Anker Signale in einem schmalen Winkel aussenden können, der über die Zeit variiert. Auf diese Weise wird ähnlich einem Leuchtturm ein rotierender Signalstrahl versendet, in dem die Position des Senders sowie der aktuelle Winkel kodiert sind. So können die Empfänger ihre Position auf eine Gerade in der Ebene (zweidimensionaler Fall) beziehungsweise auf eine Ebene im Raum (dreidimensionaler Fall) einschränken.

Empfängt ein Sensorknoten entsprechende Informationen mehrerer Anker, so kann er die jeweiligen Geraden- oder Ebenengleichungen gleichsetzen und auf

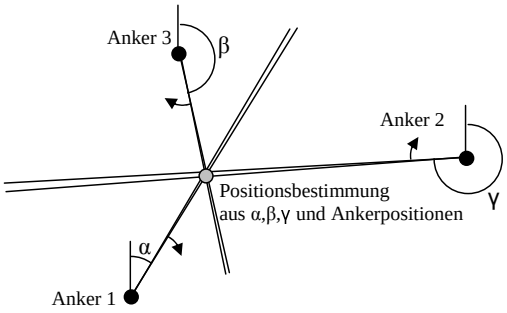


Abbildung 7.5: Positionsbestimmung mit der VOR-Methode.

diese Weise seine Position berechnen. Dieses Vorgehen wird in Abbildung 7.5 veranschaulicht, die entsprechenden Berechnungsvorschriften können [120] entnommen werden. Diese Methode verlangt, dass die Anker ihre absolute Orientierung kennen. Vorteilhafterweise ist es nicht erforderlich, dass die übrigen Sensorknoten die Richtung, aus der die Signale kommen, detektieren können.

Die Autoren von [115] schlagen Übertragung der VOR-Methode auf Sensornetze vor. Vereinfachend gehen sie aber davon aus, dass vier Anker in einem Rechteck am Rand des Sensornetzes angeordnet sind. Diese senden rotierende, in einem engen Winkel begrenzte Signale aus, die im gesamten Sensornetz empfangen werden können. Dabei können die Sensorknoten den Winkel zu den Ankern ermitteln, indem sie den bekannten Versatz der Winkel der Ankersignale nutzen, um den Empfangszeitpunkt der Signale aller Anker in einen Winkel zu überführen. Die Berechnung der Positionen der Sensorknoten kann dann lokal im Prinzip nach der oben beschriebenen Methode erfolgen, durch die Rechteckanordnung der Anker ergeben sich zusätzliche Vereinfachungen.

Die AoA-Methode hingegen erfordert, dass die Sensorknoten feststellen können, aus welcher Richtung sie die Signale der Anker empfangen, die ihrerseits omnidirektionale Sender sind. Zur Verdeutlichung des weiteren Vorgehens (hier für den zweidimensionalen Fall) sei auf Abbildung 7.6 verwiesen.

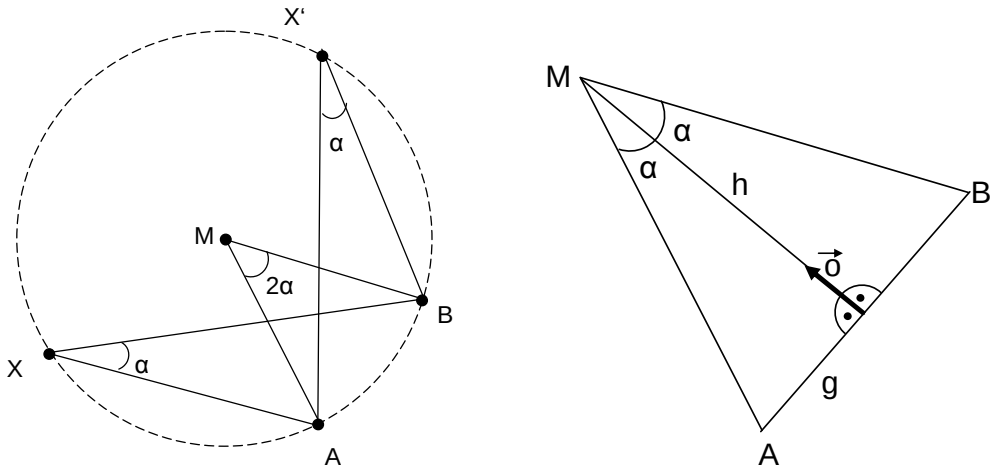
Empfängt ein Sensorknoten X entsprechende Signale von zwei Ankern A und B mit deren Positionen, so kann er den Winkel α zwischen den beiden Ankern durch Subtraktion der Empfangswinkel berechnen. Mit Hilfe von α kann X nun seine Position auf einen Kreis um den Punkt M einschränken, auf dem auch A und B liegen (Abbildung 7.6(a)). Dabei können der Punkt M wie auch der Radius des Kreises r aus den Positionen von A und B sowie dem Winkel α berechnet werden.

Es gilt

$$\vec{M} = \vec{A} + \frac{\vec{B} - \vec{A}}{2} + h\vec{o}, \tag{7.11}$$

$$h = \frac{g}{\sin\alpha}, \tag{7.12}$$

$$r = \sqrt{g^2 + h^2}, \tag{7.13}$$



(a) Einschränkung der Position von X auf einen Kreis.

(b) Berechnung von M.

Abbildung 7.6: Rückführung der Multiangulation auf die Multilateration.

wobei $\vec{o}$ den auf $\overline{AB}$ senkrecht stehenden Vektor der Länge 1 bezeichnet. Somit erhält X die Information, dass X sich in der Distanz r von Punkt M befindet. Mit Hilfe von weiteren Ankern kann auf gleiche Weise der Abstand zu weiteren Punkten ermittelt werden. Die Positionsberechnung von X erfolgt dann mittels Multilateration.

Ein besonderer Vorteil von Angle-of-Arrival ist, dass es neben der Positionsbestimmung auch die Ermittlung der Ausrichtung der Sensorknoten ermöglicht. Dann kann beispielsweise die Position eines Objektes weiter eingeschränkt werden, das mittels eines gerichteten Sensors detektiert wurde.

Das in [118] vorgestellte Verfahren zur Positionsermittlung nutzt die soeben beschriebene Vorgehensweise. Um darüber hinaus auch dann Positionen ermitteln zu können, wenn sich nicht ausreichend viele Anker in direkter Nachbarschaft befinden, wird ein *DV-Bearing* genannter Ansatz beschrieben, der die Bestimmung von Winkeln über mehrere Hops hinweg ermöglicht.

Ein zentralisierter Ansatz wird in [10] vorgestellt. Hier kommt ebenfalls Angle-of-Arrival zum Einsatz, allerdings werden alle Winkelmessungen zu einem großen Gleichungssystem zusammengefasst und zentral die resultierenden Positionen berechnet. Die Betrachtung eines Systems von Winkeln findet ihren Ausdruck auch in der Verfahrensbezeichnung *Robust System Multiangulation.*

Insgesamt konnten sich Ansätze zur Positionsbestimmung auf Basis der Multiangulation nur sehr bedingt durchsetzen, was sich neben der relativ geringen Anzahl der Publikationen auch an der minimalen Anzahl von vorgestellten praktischen Realisierungen ablesen lässt. Ursächlich ist neben dem höheren mathematischen Aufwand durch den zusätzlichen Berechnungsschritt bei der Rückführung auf die Multilateration die hohe technische Komplexität geeigne-

ter Sender und Empfänger. Obwohl die für den VOR-Ansatz notwendigen rotierenden schmalwinkligen Sender prinzipiell auch ohne mechanisch bewegliche Teile umsetzbar wären, sind keine entsprechenden Realisierungen für drahtlose Sensornetze bekannt. Für den Angle-of-Arrival-Ansatz gibt es zwar prototypische Umsetzungen, diese benötigen jedoch mehrere Empfangseinheiten pro Gerät. Das wirkt sich negativ auf Energieverbrauch, Preis und Baugröße der Geräte aus, so dass die Multiangulation speziell im Vergleich mit der Multilateration gravierende Nachteile aufweist.

Multimodale Verfahren

Aufbauend auf den beschriebenen Verfahren der Multilateration und Multiangulation gibt es in der Literatur eine Reihe von Vorschlägen, wie beide Ansätze kombiniert werden können. Die Autoren nennen neben besonderer Flexibilität gute Genauigkeiten schon bei einer geringen Anzahl von Ankern als Vorteile ihrer Ansätze.

So wird in [110] gezeigt, dass bereits mit nur zwei Ankern und drei Sensorknoten die relativen Positionen aller beteiligten Geräte abgeleitet werden können. Dabei ist es nicht zwingend erforderlich, dass die Anker ihre absoluten Positionen kennen. Fehlen die entsprechenden Informationen, werden alle Positionen in einem relativen Koordinatensystem bestimmt. Dazu kommen hier Distanzschätzungen und Angle-of-Arrival-Messungen zum Einsatz. Während dieses Verfahren bereits verteilt arbeitet, präsentierten die Autoren in [111] noch einen ähnlichen, zentralen Ansatz.

Auch in [49] wird betont, dass im Vergleich zu [121] eine ähnliche Genauigkeit bei weniger Ankern erreicht werden kann. Hier wird der VOR-Ansatz mit Signalstärkemessungen kombiniert. Zur Positionsbestimmung aus diesen unterschiedlichen Eingangsgrößen kommen sogenannte *Bayessche Netze* zum Einsatz, die wiederum auf bekannte Lokationsverfahren zurückgreifen.

Die Autoren von [119] betonen primär die besondere Flexibilität ihres Ansatzes, so dass das *Local Positioning System (LPS)* genannte Verfahren auf Geräten mit unterschiedlichen technischen Vorraussetzungen im Hinblick auf die Positionsbestimmung verwendet werden kann: LPS kann Anker mit bekannten Positionen, Kompassdaten, Distanzschätzungen und Angle-of-Arrival-Messungen flexibel für die Positionsbestimmung nutzen.

Insgesamt haften den kombinierten Verfahren inhärent die Nachteile der Multiangulation an. Erschwerend kommt die durch die Kombination höhere Komplexität der Verfahren hinzu.

7.2 Distanzschätzverfahren

Im Folgenden wird ein Überblick über aus der Literatur bekannte Verfahren zur Distanzschätzung gegeben. Dabei werden Funktionsweise und Eigenschaften der unterschiedlichen Ansätze beschrieben sowie problematische Aspekte hervorgehoben.

7.2.1 Anforderungen an Distanzschätzverfahren

In der Vergangenheit wurden verschiedenste Techniken entwickelt, die der Ermittlung von Distanzen zwischen Geräten in einem verteilten System dienen. Ihnen gemein ist, dass sie nicht direkt Distanzen messen. Vielmehr misst man eine physikalische Größe, die dann mittels eines Modells in eine Distanz umgerechnet werden. Diese Tatsache ist auch der Grund dafür, dass häufig nicht von Distanzmessung, sondern von Distanzschätzung oder Distanzermittlung gesprochen wird. Die Genauigkeit der ermittelten Distanzen hängt maßgeblich von den Eigenschaften und Eignung des verwendeten Modells ab.

Neben theoretischen Betrachtungen, die sich häufig auf die Genauigkeit der ermittelten Distanzen konzentrieren, spielt die technische Eignung für den Einsatz in realen Sensornetzen eine wesentliche Rolle. Es gibt eine Reihe von Anforderungen, die ein Verfahren zur Distanzschätzung erfüllen sollte, um sich gut für den praktischen Einsatz zu eignen.

- Verzicht auf Spezialhardware: Einige Verfahren erfordern den Einsatz eigener Hardwarekomponten speziell für die Distanzmessung. Jeder Sensorknoten muss mit den entsprechenden Funktionseinheiten ausgestattet sein. Ein Beispiel hierfür sind Ultraschallsensoren, wie sie auch im Automobil für Einparkhilfen zum Einsatz kommen. Derartige zusätzliche Hardware wirkt sich negativ auf Baugröße, Entwicklungs- und Herstellungskosten sowie den Energieverbrauch der Sensorknoten aus.
- Keine speziellen Anforderungen an die zum Einsatz kommende Hardware: Einige Verfahren erfordern zwar keine dedizierte Hardware für die Distanzmessung, stellen aber spezielle Anforderungen an die ansonsten auf den Sensorknoten zum Einsatz kommenden Komponenten. Beispiele sind eine erforderliche hohe Rechenleistung oder spezielle Eigenschaften des Funkchips. Folgen sind auch hier negative Auswirkungen insbesondere auf Preis und Energieverbrauch der Knoten.
- Geringer Energieverbrauch: Um einem Sensornetz im realen Einsatz eine lange Lebensdauer zu verleihen, ist es zwingend erforderlich, sparsam mit der zur Verfügung stehenden Energie umzugehen. Dieses gilt natürlich auch für Verfahren zur Distanzermittlung. Es ist somit geboten, Nachrichtenaustausch und Berechnungsaufwand so weit wie möglich zu begrenzen.
- Geringe Dauer der Distanzermittlung: Um ein Verfahren zur Distanzermittlung auch in mobilen Sensornetzen einsetzen zu können ist es erforder-

lich, dass die Distanzermittlung nur einen kurzen Zeitraum in Anspruch nimmt. Ist dies nicht der Fall, kann sich die Situation im Sensornetz bei Abschluss der Schätzung bereits soweit verändert haben, dass die Ergebnisse nicht mehr mit der tatsächlichen Situation übereinstimmen.

- Verzicht auf manuelle Kalibrierung: Da Sensornetze für den Einsatz ohne manuelle Konfiguration vorgesehen sind, sollten auch die Verfahren zur Distanzermittlung auf Konfiguration oder Kalibrierung durch den Nutzer verzichten. Insbesondere eine geräteindividuelle Kalibrierung ist aufgrund der avisierten Netze mit hunderten und mehr Geräten nicht akzeptabel.
- Reichweite: Die Verfahren sollten in der Lage sein, möglichst große Distanzen korrekt zu schätzen. So können längere Messabschnitte durch weniger Einzelmessungen abgedeckt werden. Auf diese Weise kann die Multi-hop-Distanzschätzung seltener zum Einsatz kommen, durch sie bedingte Fehler bekommen geringeren Einfluss.
- Robustheit gegen Variationen der Hardware oder der Umwelt: Da Sensornetze zum Teil unter a priori unbekannten Umgebungsbedingungen zum Einsatz kommen, ist es wichtig, dass die ermittelten Distanzen möglichst robust gegen äußere Einflüsse sind. Auch Toleranzen der zum Einsatz kommenden Hardware sollten die Ergebnisse möglichst wenig negativ beeinflussen. Ebenfalls wichtig ist, dass ein Verfahren möglichst immer zu einer Distanzschätzung gelangt.

7.2.2 Überblick über Distanzschätzverfahren

Die verschiedenen Verfahren zur Distanzermittlung lassen sich anhand der zugrunde liegenden Funktionsweise in Gruppen einteilen. Jede einzelne Technik basiert auf unterschiedlichen Hardwarekomponenten, Messtechniken und Modellen, um autonom Distanzen zwischen Geräten zu ermitteln. Als Folge dessen ergeben sich unterschiedliche Eigenschaften und Anforderungen der vielfältigen Verfahren. Sieben verschiedene Prinzipien sind heute bekannt:

- Signallaufzeit
- Differenzielle Signallaufzeit
- Einstrahldauer
- Signalabschwächung
- Konnektivität
- Interferometrie
- Signalumlaufzeit

Das vielleicht prominenteste Verfahren ist die Messung von Distanzen auf Basis von *Signallaufzeiten*. Es kommt beispielsweise bei GPS [75] zum Einsatz. Die Grundidee der Distanzermittlung ist hier, dass ein Sender ein Signal aussendet, dessen Aussendezeitpunkt dem Empfänger bekannt sein muss. Der Empfänger ermittelt den Zeitpunkt, zu dem das Signal ihn erreicht, und kann aus der

Differenz der beiden Zeitpunkte die Signallaufzeit errechnen. Ist die Ausbreitungsgeschwindigkeit des Signals bekannt, kann die Distanz zwischen Sender und Empfänger ermittelt werden. Problematisch an diesem Ansatz ist, dass der Zeitpunkt des Aussendens dem Empfänger bekannt sein muss. Dieses erfordert eine sehr genaue Zeitsynchronisation zwischen beiden Geräten: Fehler in der Synchronisation führen direkt zu Fehlern in der Laufzeitberechnung.

Das Verfahren hat den Vorteil, dass keine spezielle Hardware benötigt wird, wenn Funksignale zum Einsatz kommen. Aufgrund der hohen Ausbreitungsgeschwindigkeit wäre allerdings eine extrem genaue Synchronisation erforderlich. Dies führt dazu, dass das Verfahren in Sensornetzen praktisch nicht verwendbar ist.

Abhilfe versprechen Verfahren, die auf dem Prinzip der Messung der *differenziellen Signallaufzeit* beruhen. Beispiele hierfür sind Calamari [176], Cricket [131], AHLoS [144] und andere [64,70,141,177]. Hier werden zwei Signale ausgesandt, die sich mit unterschiedlichen Geschwindigkeiten ausbreiten. Kennt man diese Ausbreitungsgeschwindigkeiten, kann man aus der Zeitdifferenz zwischen der Ankunft beider Signale die Distanz zwischen Sender und Empfänger errechnen.

Die meisten der Systeme setzen hörbaren oder Ultraschall einerseits und Funksignale andererseits ein. Alleinige Messungen der Empfangsintervalle liefern dabei zunächst Fehler von etwa 74% [176]. Techniken der Signalverarbeitung wie Rauschunterdrückung, digitale Filterung, Maximalwertbestimmung und Kalibrierung ermöglichen jedoch erheblich höhere Genauigkeiten. Während einige Autoren mittlere Fehler von 10% ermittelten [176], berichten andere von Fehlern von etwa 1% bei einer maximalen Distanz von 9 m [141]. Die Autoren von [177] erzielten Reichweiten von 12 m bei einem Fehler von 0.5%.

Diese Systeme liefern recht genaue Ergebnisse, doch dieser Vorteil ist teuer erkauft. Da sie auf Basis differenzieller Laufzeitmessungen arbeiten, benötigen sie inhärent einen zusätzlichen Sender und Empfänger. Dies wirkt sich negativ auf Entwicklungs- und Herstellungskosten, Baugröße und Energieverbrauch von Sensorknoten aus, die solche Systeme verwenden. Darüber hinaus bringt der Einsatz von Schall zusätzliche Einschränkungen mit sich. So führen lokale Temperaturdifferenzen zu variierenden Ausbreitungsgeschwindigkeiten. Hindernisse in der direkten Sichtlinie können darüber hinaus dazu führen, dass die Laufzeitmessung verfälschte oder gar keine Ergebnisse liefert. Auch sind Schallabstrahlung und -wahrnehmung eines jeden Emitters beziehungsweise Detektors auf einen bestimmten Winkel begrenzt, so dass jeweils mehrere zum Einsatz kommen müssen um omnidirektionale Messungen zu ermöglichen. Weiterhin benötigen Sender und Empfänger überproportional viel Energie. Der Einsatz von Schall limitiert auch die maximale Reichweite des Messsystems. Sie liegt typischer Weise bei 3–15 m [144], d.h. sie erreicht nur einen Bruchteil der Reichweite der Funkschnittstelle. Neben der Kalibrierung ist die geeignete Verarbeitung der empfangenen Signale problematisch. Variiert die Verarbeitungszeit von Messung zu Messung, so werden die Ergebnisse verfälscht. Vorteilhaft ist, dass die Distanzschätzung nur kurze Zeit in Anspruch nimmt.

Eine Sonderstellung nimmt die Distanzschätzung auf Basis der *Einstrahldauer* ein. Sie richtet sich an eine spezielle Unterkategorie der Sensornetze, die das Laserlicht einer zentralen Basisstation zur Kommunikation nutzt [87, 174]. Bei diesen Netzen leuchtet ein Laserstrahl die Sensorknoten nacheinander an. Diese können einerseits den Laser detektieren und ihn andererseits mittels dreier senkrecht zueinander stehender Spiegel reflektieren. Da einer der drei Spiegel beweglich ist, können die Sensorknoten das Laserlicht in zeitlicher Abfolge zurückwerfen oder nicht, und so ein Signal auf das zurückgeworfene Licht modulieren.

Das in [140] vorgestellte Verfahren nutzt die Fähigkeiten der Sensorknoten zur Laserdetektion aus. Es basiert auf einer Art rotierendem Leuchtturm, der einen horizontal perfekt parallelen und vertikal sehr breiten Laserstrahl aussendet. Ein lasersensibler Sensorknoten nimmt diesen rotierenden Laser als ein kurzes Aufblitzen wahr, wenn er von dem vorbeiwandernden Strahl angeleuchtet wird. Die Einstrahldauer des Lasers hängt dabei von der Distanz des Sensorknotens zum Leuchtturm ab. Somit kann der Knoten aus der Dauer der Beleuchtung und der a priori bekannten Drehungsgeschwindigkeit auf die Distanz zum Leuchtturm schließen.

Der in der Publikation beschriebene Prototyp erreichte eine Genauigkeit von etwa 2% bei einer maximalen Reichweite von 14 m, die sich allerdings mittels einer ausgereifteren Mechanik auf 120 m bis 140 m erweitern ließe. Der Hauptnachteil des Verfahrens ist, dass zur Distanzschätzung eine ununterbrochene Sichtverbindung zwischen Sensorknoten und Basisstation erforderlich ist. Andernfalls kommt keinerlei Schätzung zu Stande. Vorteile des Verfahrens sind, dass in Sensornetzen, die ohnehin auf Basis von Laserlicht kommunizieren, keine zusätzliche Hardware erforderlich ist. Die Verarbeitung der entsprechenden Signale sowie die Signalverarbeitung sind einfach und verursachen nur geringen Rechenaufwand. Die heute üblichen Sensorknoten mit Funkkommunikation hingegen benötigen zusätzlich eine spezielle Laserdetektionseinheit, um dieses Verfahren anwenden zu können. Das wirkt sich negativ auf den Energieverbrauch, den Preis und die Baugröße aus.

Distanzschätzungen auf Basis der Funkschnittstelle

Um den Nachteilen der Systeme, die auf der Messung von (differenzieller) Laufzeitmessung basieren, aus dem Weg zu gehen, wurden andere Verfahren zur Distanzschätzung entwickelt. Dabei stand die Beschränkung auf die Nutzung der Funkschnittstelle, mit der die Geräte ohnehin ausgerüstet sind, im Mittelpunkt.

Diese Ansätze lassen sich in zwei Kategorien unterteilen:

- Qualitative Verfahren (range-free): Die hier eingeordneten Verfahren prüfen lediglich, ob der Empfang der Signale eines bestimmten Empfängers möglich ist. Falls ja, leitet man daraus ab, dass die Distanz unterhalb der a priori bekannten maximalen Reichweite liegt.

- Quantitative Verfahren (range-based): Verfahren in dieser Gruppe versuchen, aus Signaleigeschaften, die an der Funkschnittstelle messbar sind, konkrete Entfernungen zwischen Sender und Epfänger abzuleiten.

Die ersten Ansätze zur Distanzschätzung im Bereich des Ubiquitous Computing gehörten zur Kategorie der *qualtitativen Verfahren*. Sie beschränkten sich auf die Detektion räumlicher Nähe. Beispiele für diese Art der Distanzschätzung sind das „Active Badge“ System [173] und das „Hybrid indoor navigation system“ [35]. Beide nutzen infrarotes Licht, um Beacons mit einer eindeutigen Identifikation auszusenden. Die Empfänger dieser Signale – entweder Teile einer fest installierten Infrastruktur oder aber die mobilen Geräte – leiten daraus räumliche Nähe zum Sender der Beacons ab, da die Reichweite des Infrarotsignales auf einige Meter begrenzt ist. Mit einer großen Anzahl dicht angeordneter Sender lässt sich auf diese Weise eine recht feine Auflösung darstellen. Die wesentlichen Nachteile sind der Bedarf an einer festen Infrastruktur und der immense Installationsaufwand.

Um diese Technik überhaupt in Verbindung mit Sensornetzwerken einsetzen zu können, wurden verschiedene Anstrengungen unternommen qualitative Verfahren so zu gestalten, dass sie ohne feste Infrastruktur auskommen. Dazu wurde angenommen, dass es ausgezeichnete Knoten – so genannte Anker – gibt, zu denen Abstände bestimmt werden. Diese senden mittels der vorhandenen Funkschnittstelle spezielle Funksignale aus, deren Empfänger auf die Nähe eines Ankers schließen können [28, 156]. Geblieben ist der Nachteil, dass Distanzen unterhalb des Kommunikationsradius nicht differenziert werden können. Daher sind die resultierenden Abstandsschätzungen eher grobgranular, unterhalb des Kommunikationsradius beschränken sie sich auf die Erkennung von Nähe. Verbesserungen der Granularität werden möglich, indem man so viele Anker einsetzt, dass jeder Knoten in Hörweite von mehreren Ankern liegt [28, 156]. Andere Verfahren beschränken sich auf das Zählen von Hop-Abständen und werden daher im sich anschließenden Abschnitt über Multi-hop-Distanzschätzung besprochen.

Um differenziertere und feingranularere Aussagen über Distanzen treffen zu können, wurden unterschiedliche *quantitativen Verfahren* entwickelt.

Ein Verfahren, um Distanzschätzungen auf Basis von *Konnektivitätsinformationen* tätigen zu können, wird in [31, 32] vorgestellt. Das *Neighborhood Intersection Distance Estimation Scheme (NIDES)* basiert auf der Idee, dass nahe beieinander liegende Knoten im Allgemeinen mehr gemeinsame Nachbarn haben als weiter voneinander entfernte.

Es wird dargelegt, wie aus dem Anteil gemeinsamer Nachbarn zweier Knoten $s \in [0, 1]$ die Distanz d zwischen den beteiligten Knoten abgeschätzt werden kann. Die Distanz wird durch eine Funktion $d(s)$ beschrieben, die die Umkehrfunktion der sogenannten Schätzfunktion $s(d)$ ist, es gilt also

$$d(s) = s^{-1}(d). \tag{7.14}$$

Da die Funkeigenschaften einen wichtigen Einflussfaktor bezüglich der Distanzschätzung darstellen, stellt ein Modell dieser Eigenschaften den Ausgangspunkt zur Bestimmung der Schätzfunktion $s(d)$ dar. Es wird durch eine Funktion $f : \mathbb{R} \times \mathbb{R} \rightarrow [0, 1]$ beschrieben. Sie gibt die Wahrscheinlichkeit an, dass ein Knoten, der an der Position (x, y) liegt, Funksignale eines im Ursprung positionierten Knotens empfangen kann.

Der eigentliche Berechnungsschritt wird nun zunächst anschaulich beschrieben. Zur Vereinfachung wird hierzu eine rotationssymmetrische Funktion f herangezogen. Der voraussichtliche Anteil gemeinsamer Nachbarn s bei einer konkreten Distanz d lässt sich bestimmen, indem man f um d entlang der x-Achse verschiebt, sie mit ihrer nicht verschobenen Ausgangsfunktion multipliziert und das resultierende Volumen des Ergebnisses ins Verhältnis zum Volumen von f setzt, also auf das Volumen von f normiert. Dieses ist in Abbildung 7.7 beispielhaft für das Kreismodell dargestellt, das Ergebnis der Multiplikation ist grau markiert. Um einen funktionalen Zusammenhang zwischen der Verschiebung (oder auch Distanz) d und dem Anteil der gemeinsamen Nachbarn s zu erhalten, muss man diesen Vorgang für alle d durchführen. Diesen Prozess nennt man Faltung (die abschließende Normierung ist allerdings nicht Teil der Faltung).

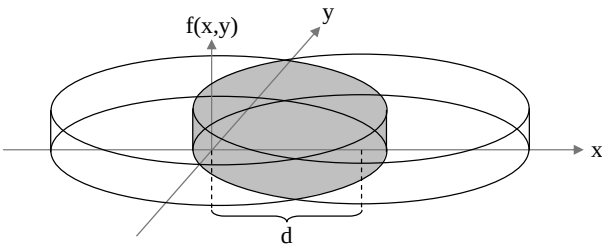


Abbildung 7.7: Veranschaulichung einer Faltung

Die Faltung von f entlang der x-Achse wird von der Gleichung

$$(f * f)(d) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x - d, y)\ f(x, y)\, \mathrm{d}x\, \mathrm{d}y \tag{7.15}$$

beschrieben. Um das Faltungsvolumen ins Verhältnis zum Volumen unterhalb von f zu setzen, muss man auf dieses Ursprungsvolumen normieren. Nimmt man die Normierung hinzu, ergibt sich

$$s(d) = \frac{\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x - d, y) f(x, y)\, \mathrm{d}x\, \mathrm{d}y}{\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y)\, \mathrm{d}x\, \mathrm{d}y}. \tag{7.16}$$

Da im Allgemeinen f nicht rotationssymmetrisch ist, muss während der Faltung auch eine Rotation durchgeführt werden.

$$f_{rot}(x, y, \alpha) = f(x\ cos(\alpha) + y\ sin(\alpha), -x\ sin(\alpha) + y\ cos(\alpha)) \tag{7.17}$$

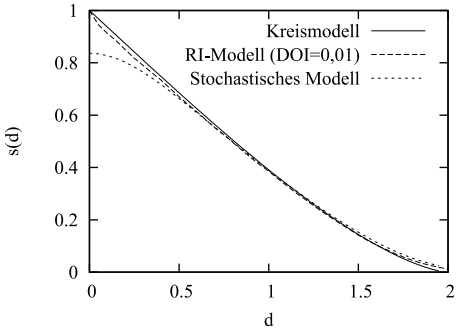
gibt eine Funktion an, die aus f durch Drehung um die z-Achse um den Winkel α entsteht. Durch Subtraktion von d wird sie entlang der x-Achse um d verschoben, d entspricht dabei dem Abstand zwischen Sender und Empfänger.

Der Anteil gemeinsamer Nachbarn über der Distanz lässt sich somit durch $s : \mathbb{R}^+ \to [0, 1]$ beschreiben:

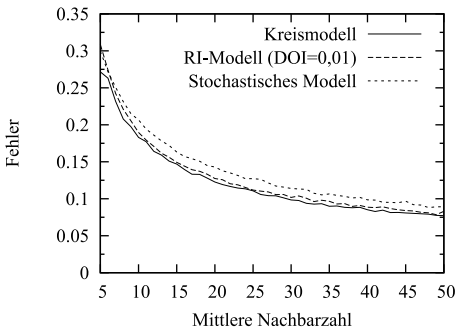
$$s(d) = \int_0^{2\pi} \frac{\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f_{rot}(x - d, y, \alpha)\ f_{rot}(x, y, \alpha)\, \mathrm{d}x\, \mathrm{d}y}{\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f_{rot}(x, y, \alpha)\, \mathrm{d}x\, \mathrm{d}y} \mathrm{d}\alpha. \tag{7.18}$$

Die Funktion der Distanz in Abhängigkeit von s ergibt sich dann mit Hilfe von 7.14.

Trotz ihrer komplizierten Berechnung kann $d(s)$ auch auf ressourcenbegrenzten Geräten problemlos zum Einsatz kommen, da dieser Rechenschritt im Vorfeld des Deployments erfolgen kann, und als kompakte Tabelle repräsentiert werden kann. Abbildung 7.8(a) zeigt die resultierenden Verläufe von $s(d)$ für unterschiedliche Funkmodelle. Dabei ist die Distanz auf den mittleren Kommunikationsradius r normiert. Das Protokoll, das das hier beschriebene Verfahren umsetzt, wurde in [31] im Detail beschrieben. Es benötigt lediglich 2 versendete Nachrichten pro Gerät, weniger als 50 Byte Speicher und ist auch auf einfachsten Controllern implementierbar.



(a) Schätzfunktion $s(d)$



(b) Schätzfehler in Abhängigkeit von der mittleren Anzahl von Nachbarn

Abbildung 7.8: Schätzfunktion und Schätzfehler für verschiedene Funkausbreitungsmodelle.

Die Genauigkeit der Schätzungen wurde nur mit Hilfe von Simulationen mit dem Netzwerksimulator ns-2 [162] überprüft. Abbildung 7.8(b) zeigt den Schätzfehler für verschiedene Ausbreitungsmodelle in Abhängigkeit von der Netzdichte, also der mittleren Anzahl von Nachbarn. Dabei ist der Fehler als Bruchteil des Kommunikationsradius r angegeben. Es ist zu erkennen, dass es kaum Unterschiede zwischen den verschiedenen Funkausbreitungsmodellen gibt. Der mittlere Schätzfehler nimmt mit zunehmender Netzdichte kontinuierlich ab. Er liegt bei etwa 20% des mittleren Kommunikationsradius bei einer Dichte von 10 und fällt auf 8% des mittleren Kommunikationsradius bei einer Dichte von 50 ab.

Das bedeutet, dass die Schätzungen mit zunehmender Dichte immer genauer werden.

Viele andere Publikationen schlagen vor, zur Abstandsschätzung die *Abschwächung der Funksignale*, die Sensorknoten im Rahmen ihrer Kommunikation austauschen, zu nutzen. Die meisten Funkinterfaces stellen zu diesem Zweck einen numerischen Wert zur Verfügung, der die Signalstärke der empfangenen Funkdaten und somit indirekt die Verbindungsqualität widerspiegelt. Dieser Wert wird daher i.A. *Received Signal Strength Indicator* (RSSI) oder *Link Quality Indicator* (LQI) genannt, im Folgenden wird der erste der beiden Begriffe stellvertretend für alle derartigen Verfahren verwendet.

Systeme, die Distanzschätzungen auf Basis des RSSI-Wertes durchführen [28, 36, 142], liefern für kurze Distanzen recht genaue Werte, wenn umfangreiche Nachverarbeitung der einzelnen Werte erfolgt. Für größere Distanzen sind die Ergebnisse jedoch ungenau [103].

Bei Distanzen von weniger als 20 m liefern einfache Funkchips Fehler zwischen 50 und 100 Prozent ihrer Kommunikationsreichweite, während aufwändigere Modelle den Fehler hier auf etwa 10 % begrenzen können, letzteres jedoch nur nach aufwendiger in-situ Kalibrierung [177]. Negativ auf die Genauigkeit wirken sich Mehrwegeausbreitung, Interferenz und Abschattung durch Hindernisse aus. Diese Effekte erschweren die Entwicklung eines konsistenten Modells zur Umsetzung der gemessenen Werte in Distanzen [144]. Die Folge ist, dass Verfahren, die ausschließlich auf Signalabschwächung beruhen, lediglich mittelmäßige Genauigkeiten erreichen [50].

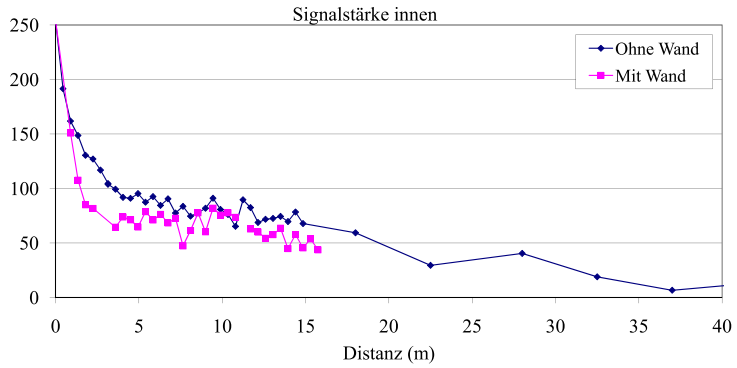


Abbildung 7.9: Messung der Signalstärke über die Entfernung.

Weitere Einschätzungen hinsichtlich der Möglichkeiten und Grenzen von Distanzschätzungen auf Basis von Signalstärkenmessungen werden in [34] präsentiert. Abbildung 7.9 zeigt das Ergebnis einer in einem Bürogebäude aufgenommenen Signalstärkenmessreihe. Sie ist entstanden, indem zwei Geräte in verschiedenen Distanzen zueinander positioniert werden, während sie kontinuierlich Daten austauschen. Die dabei gemessenen Signalstärken (einheitenlos an-

gegeben) werden gespeichert und können später den entsprechenden Distanzen zugeordnet werden.

Die dunklere Kurve zeigt Messungen zwischen Geräten mit direktem Sichtkontakt, dabei entspricht jeder Punkt dem Mittelwert der für die entsprechende Distanz gemessenen Signalstärken. Es ist zu erkennen, dass grundsätzlich die gemessene Signalstärke wie erwartet mit wachsender Distanz abnimmt. Allerdings ist der Abfall der Signalstärke nicht kontinuierlich. Während die Kurve im Bereich unterhalb von 5 Metern steil verläuft, flacht sie mit zunehmender Distanz ab und steigt zwischenzeitlich immer wieder an. Diese „Buckel“ entstehen, wenn Signale an Wänden reflektiert werden und sich mit den auf direktem Weg angekommenen Signalen überlagern. Dadurch kann die Amplitude des Signals verstärkt oder abgeschwächt werden, und es kommt zu den erkennbaren Schwankungen. Die hellere der beiden Kurven zeigt eine vergleichbare Messreihe, die allerdings zusätzlich durch eine Wand hindurch aufgenommen wurde.

Es ist ebenfalls zu erkennen, dass die Dämpfung einer zwischen Sender und Empfänger befindlichen Wand die gemessene Signalstärke reduziert. Weitere Messungen haben gezeigt, dass die Abschwächung von Materialart und -stärke abhängt, es lassen sich also bei bekannten Wandeigenschaften die resultierenden Dämpfungen analytisch ableiten. Es wird jedoch anhand der Abbildung auch deutlich, dass die beiden Kurven teilweise einen sehr ähnlichen Verlauf zeigen, und somit aus Empfängersicht teils schwer zu unterscheiden wären.

Insgesamt zeigt sich, dass die Signalstärke nur bedingt zur Distanzschätzung geeignet ist. Der flache Verlauf der Kurve bei größeren Distanzen in Kombination mit durch Dämpfung und Reflektion entstehenden Variationen macht eine genaue Bestimmung der Distanz zwischen zwei Geräten in Gebäuden schwierig. Es ist somit mit nicht unerheblichen Fehlern der ermittelten Distanzen zu rechnen.

Den Nachteilen mittelmäßiger Genauigkeit und der erforderlichen Kalibrierung stehen verschiedene Vorteile gegenüber. So wird keine zusätzliche Hardware benötigt, da die bestehende Funkschnittstelle genutzt wird. Das Verfahren arbeitet schnell und benötigt nur wenig Energie, wenn die ohnehin anfallende Kommunikation zur Messung der RSSI-Werte herangezogen wird. Prinzipiell können Distanzen bis zum Kommunikationsradius geschätzt werden, mit zunehmender Distanz nimmt die Genauigkeit jedoch ab.

In [103] wird ein *Radiointerferometrie* genanntes Verfahren vorgestellt, das sehr gute Genauigkeiten liefert. Es basiert auf der Idee, dass verschiedene Knoten, die Sinussignale leicht unterschiedlicher Frequenz abstrahlen, bei einem Empfänger in Abhängigkeit von der Distanz in verschiedenen Intervallen an- und abschwellende Überlagerungsergebnisse erzeugen. Kennt man mehrere solcher Intervalle, kann man ohne hochgenaue Zeitsynchronisation Rückschlüsse auf die Distanzen ziehen.

Maroti et. al berichten von einer mittleren Abweichung von 3 cm und einer Reichweite von 160 m. Dabei betrug der maximal festgestellte Fehler 6 cm, was etwa 0,04% entspricht.

Dieser hervorragenden Genauigkeit stehen verschiedene Nachteile gegenüber. Obwohl prinzipiell die vorhandene Funkschnittstelle genutzt werden kann, kann das Verfahren nur mit speziellen Funkchips zur Anwendung kommen, die die Emission von Sinuswellen dicht beieinander liegender Frequenzen unterstützen. Diese sind zumindest zurzeit eher die Ausnahme am Markt. Die hohe algorithmische Komplexität stellt darüber hinaus hohe Anforderungen an die Rechenleistung der Sensorknoten. Des Weiteren sind die Messungen langwierig, wodurch sich das Verfahren nicht für mobile Szenarien eignet. Während dessen wird der Funkkanal belegt, was sich negativ auf den sonstigen Betrieb des Sensornetzes auswirkt und darüber hinaus viel Energie verbraucht. Wie bei den meisten Verfahren, die auf Funkmessungen beruhen, ist eine recht große Anfälligkeit gegen Einflüsse von Mehrwegeausbreitung, Interferenz und Abschattung durch Hindernisse zu erwarten. Diese wurde jedoch in [103] nicht untersucht.

Eine weitere Alternative stellt die Distanzmessung auf Basis von *Signalumlaufzeiten* dar. Dabei versendet ein Gerät ein Signal und startet eine Zeitmessung. Das empfangende Gerät versendet sofort eine Antwort. Kommt diese beim ursprünglichen Absender an, stoppt dieser die Zeitmessung. Aus der Gesamtlaufzeit kann dann bei bekannter Ausbreitungsgeschwindigkeit der Signale der zurückgelegte Weg ermittelt werden. Die Schwierigkeit besteht hier darin, dass sich die Signale mit Lichtgeschwindigkeit bewegen. Somit ist eine extrem feingranulare Zeitmessung erforderlich, die technisch hoch anspruchsvoll ist. Diese Technologie wurde erst in der letzten Zeit so weit entwickelt, dass sie in einen einzigen Funkchip integriert erhältlich ist. Somit ist der Kreis der Funkchips, die diese Technik beherrschen, zurzeit noch sehr eingeschränkt.

Im Rahmen von [34] werden Messungen mit dem damals einzigen lieferbaren System dieser Art (Nanotron Nanoloc) durchgeführt.

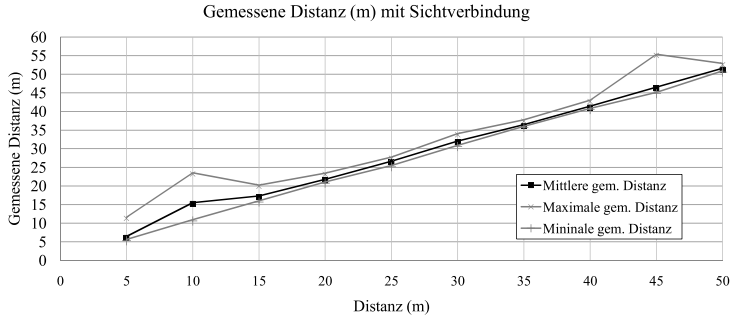


Abbildung 7.10: Distanzmessung mittels Signalumlaufzeit bei Sichtverbindung.

Abbildung 7.10 zeigt den Zusammenhang zwischen tatsächlicher und mittlerer gemessener Distanz (sowie gemessene Minima und Maxima) unter idealen

Bedingungen. Das bedeutet insbesondere, dass die beiden Geräte Sichtkontakt haben. In dieser Situation sind die Messwerte nahezu perfekt, tatsächliche und gemessene Distanzen weichen praktisch nicht voneinander ab.

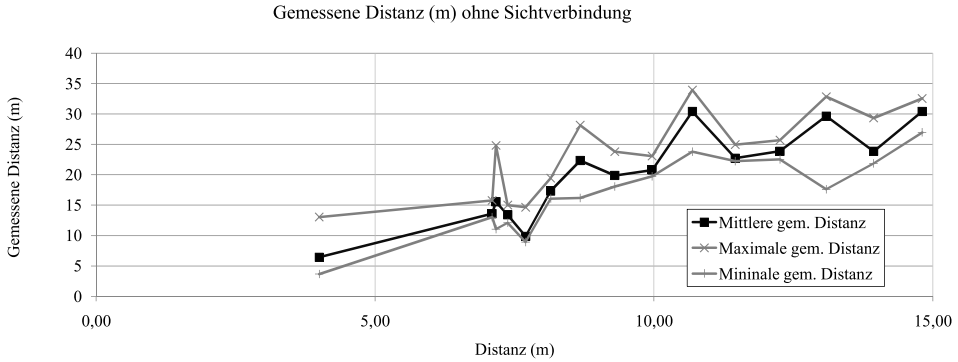


Abbildung 7.11: Distanzmessung mittels Signalumlaufzeit ohne Sichtverbindung.

Allerdings ist Sichtkontakt bei den meisten Anwendungen eher die Ausnahme. Daher wurde die Messgenauigkeit auch unter weniger günstigen Bedingungen untersucht, ohne Sichtkontakt der Geräte (Abbildung 7.11). Hier wird deutlich, dass die Messwerte i.A. massiv über den tatsächlichen Distanzen liegen. Die Ursache ist, dass offensichtlich nicht der direkte Weg zwischen den Geräten gemessen wird. Vielmehr umrundet das Signal die in der direkten Sicht befindlichen Hindernisse und nimmt somit einen längeren Weg. Daher drückt der Messwert nicht die Distanz zwischen den Geräten, sondern die Länge des Signalweges aus.

Dieser Effekt liegt vermutlich darin begründet, dass der Empfänger nicht das zuerst eintreffende, sondern das stärkste eintreffende Signal empfängt. Somit wird nicht das direkte, durch das Hindernis abgeschwächte Signal ausgewertet, sondern das stärkere indirekte Signal.

Auf diese Weise kommt es zu massiven Distanzüberschätzungen, wenn nicht der direkte Weg genommen wurde. Unglücklicherweise ist aus Sicht der beteiligten Geräte nicht erkennbar, ob es sich um eine direktes Signal (und somit um ein sehr genaues Ergebnis) oder um eine indirektes (und somit sehr ungenaues) handelt.

Eine weitere prinzipielle Schwäche des Verfahrens ist, dass bidirektionale Kommunikation zur Distanzschätzung erforderlich ist. Es konnte jedoch gezeigt werden, dass ein durchaus nicht zu unterschätzender Anteil der Kommunikationswege lediglich unidirektional ist.

7.2.3 Multi-hop Distanzschätzungen

Allen bisher betrachteten Verfahren ist gemein, dass sich hiermit lediglich Abstände zwischen Geräten schätzen lassen, die direkt miteinander kommunizieren können. Bei etlichen Sensornetzanwendungen ist es jedoch wünschenswert, dass sich auch Abstände zwischen Geräten bestimmen lassen, die nicht direkt miteinander kommunizieren können; d.h. die Entfernung zwischen den Geräten ist größer als die Kommunikationsreichweite. Auch hierzu wurden bereits Arbeiten durchgeführt.

Der einfachste Ansatz besteht darin, die Entfernungen, die zwischen zwei direkt benachbarten Knoten geschätzt werden, über die einzelnen Hops entlang eines Multi-hop-Pfades aufzuaddieren: Damit alle Geräte die Distanz zu einem Knoten ermitteln können, sendet dieser eine Flutwelle aus. Jedes Gerät addiert beim Weiterleiten der Nachricht seine Distanz zum Vorgänger auf die in der Nachricht enthaltene Distanz auf. Lernt ein Gerät auf diese Weise einen Pfad mit einer kürzeren Distanz zum ursprünglichen Absender kennen, leitet es die Nachricht weiter, sonst verwirft es sie. Ein entsprechendes Verfahren wird in [145] beschrieben. Im Folgenden werden wir diesen Ansatz mit *Sum-Dist* bezeichnen - die Autoren von [97] verwenden diese Bezeichnung ebenfalls.

Ein Nachteil von *Sum-Dist* besteht darin, dass sich die Schätzfehler über mehrere Hops akkumulieren können: Wird bei einem Verfahren stets über- oder unterschätzt, zum Beispiel weil Umwelteinflüsse die Messungen zeitweilig beeinträchtigen, summieren sich die Schätzfehler. Niculescu und Nath schlagen in [117] *DV-Hop* vor. Bei diesem Verfahren wird lediglich die minimale Anzahl der Weiterleitungen (statt der geschätzten Abstände) durch das Netzwerk propagiert. Kennt man die mittlere Hoplänge, kann man diese mit der Anzahl der erforderlichen Hops multiplizieren und erhält so eine Abstandsschätzung. Zur Ermittlung der mittleren Hoplänge ist ein separater Kalibrierungsschritt erforderlich, bei dem man davon ausgeht, dass im Netzwerk mehrere Geräte vorhanden sind, die ihren Abstand zueinander kennen. Diese können nach Empfang der ersten Flutwelle die mittlere Hoplänge berechnen und im Netz propagieren.

Sowohl Sum-Dist und DV-Hop berücksichtigen keine geometrischen Abhängigkeiten, d.h. die geschätzten Distanzen werden nicht dahingehend überprüft, ob sie in der 2D-Ebene überhaupt geometrisch möglich sind. Diesen Aspekt greift das Verfahren *Euclidean* [117] von Niculescu und Nath auf. Hier werden geometrische Beziehungen zwischen mehreren Knoten ausgenutzt, um Distanzen so zu schätzen, dass sie in der 2D-Ebene auch möglich sind. Das Verfahren hat den Nachteil, dass nicht jeder Knoten zu einer Distanzschätzung gelangt. Eine zusammenfassende Darstellung dieses Verfahrens findet der Leser in [97].

7.2.4 Zusammenfassung

In diesem Kapitel wurde eine Vielzahl unterschiedlicher Ansätze zur Positionsbestimmung in drahtlosen Sensornetzen vorgestellt und diskutiert.

Aus Sicht der Autoren sind die Verfahren auf Basis der Multilateration am ehesten für den Einsatz in Sensornetzen geeignet. Hier bestimmen alle Knoten zunächst ihre (Multi-hop-)Distanz zu sogenannten Ankern, Sensorknoten mit bekannter Position. Aus deren Positionen und den Distanzen dorthin wird dann die Position eines jeden Sensorknotens bestimmt.

Die Verfahren auf Basis der Multilateration bieten eine Reihe von Vorteilen. Zum einen sind sie, anders als Flächenschnitte und Positionsmittelung, auch für große Multi-hop-Netze geeignet und benötigen nur eine geringe Zahl von Ankern. Zum anderen arbeiten sie im Gegensatz zu Verfahren aus dem Bereich der Constraints und Optimierungen inhärent verteilt, und vermeiden so hohes Datenaufkommen und Skalierungsprobleme zentraler Ansätze. Gegenüber der Multiangulation vermeiden Sie komplizierte Messungen, die nur mit zusätzlicher Hardware möglich wären, wenn sie mit Hilfe der Funkschnittstelle arbeiten.

Im weiteren Projektverlauf sollen daher zur Positionsbestimmung selbst die Verfahren *Lateration* und *Min-Max* genauer betrachtet werden. Dazu ist es erforderlich, die Multi-hop-Distanzen zu den Ankern zu ermitteln. Um diese aus den Single-hop-Distanzen zu bestimmen, werden die Verfahren *Eucledian*, *Sum-Dist* und *Distance-Vector-Hop* intensiver untersucht.

Allerdings nehmen voraussichtlich die Single-hop-Distanzmessungen großen Einfluss auf die Genauigkeit der Positionsberechnung. Da aus Sicht der Autoren ausschließlich Verfahren, die nur auf Basis der bereits existierenden Funkschnittstelle arbeiten, zum Einsatz in Sensornetzen geeignet sind, wurde diesen besondere Aufmerksamkeit geschenkt. Trotz aller ihrer Defizite scheinen nach der vorangegangenen Analyse insbesondere Distanzschätzverfahren mittels Signalabschwächung und Signalumlaufzeit vielversprechend.

Insbesondere in letzterem Bereich gab es kürzlich eine Neuentwicklung: der kombinierte Funk- und Controllerchip Jennic JN5148, der unter anderem auch auf dem Core Module 2 (CM20X) der coalesenses GmbH eingesetzt wird, bietet als momentan einziger zum Standard IEEE802.15.4 kompatibler Funkchip eine Distanzschätzung mit Hilfe von Signalumlaufzeitmessungen.

Obwohl im Rahmen von WSNLAB aus momentaner Sicht keine CM20X Module eingesetzt werden, sondern auf die älteren Grundmodule CM10X zurückgegriffen werden muss, wurden ebenfalls Distanzmessungen auf Basis der Signalumlaufzeit mit Hilfe des JN5148 durchgeführt, um die erzielbaren Genauigkeiten zu untersuchen. Die Ergebnisse dieser Messungen werden in Abschnitt 10.2 vorgestellt.

8 Simulative Evaluation von Lokalisierungsverfahren

Dieses Kapitel widmet sich der simulativen Evaluation verschiedener Lokalisierungsverfahren. In Kapitel 7 wurde bereits eine Vielzahl unterschiedlicher Ansätze zur Positionsbestimmung in drahtlosen Sensornetzen vorgestellt und diskutiert. Die Verfahren *Lateration* und *Min-Max* wurden dabei als besonders geeigent identifiziert. Zur Positionsbestimmung einzelner Knoten müssen diese jedoch Multi-hop-Distanzen zu den Ankern ermitteln. Um diese aus den Single-hop-Distanzen zu bestimmen, werden die Verfahren *Euclidean*, *Sum-Dist* und *Distance-Vector-Hop* intensiver untersucht.

Von großer Bedeutung für die Genauigkeit der Single- und Multi-hop-Distanzschätzung sind die Single-hop-Distanzmessungen. Da aus Sicht der Autoren ausschließlich Verfahren, die nur auf Basis der bereits existierenden Funkschnittstelle arbeiten, zum Einsatz in Sensornetzen geeignet sind, wurde diesen besondere Aufmerksamkeit geschenkt. Trotz aller ihrer Defizite scheinen nach der vorangegangenen Analyse insbesondere Distanzschätzverfahren mittels Signalabschwächung und Signalumlaufzeit vielversprechend. In der Vergangenheit wurden Distanzmessungen mit Hilfe des im Projekt verwendeten JN5139 durchgeführt, um die erzielbaren Genauigkeiten zu untersuchen. Diesen Messungen kommt eine zentrale Bedeutung bei der simulativen Untersuchung der Genauigkeit der verschiedenen Verfahren zu. Innerhalb einer Simulationsumgebung müssen die Distanzmessungen zwischen zwei Knoten auf geeignete Art und Weise simuliert werden.

Im Rahmen dieses Projektes wurde zunächst aus Messergebnissen ein Distanzschätzungsmodell erstellt. Diese Modellbildung wird in Abschnitt 8.1 detailliert beschrieben. Abseits der verwendeten Verfahren beeinflusst auch das konkrete Szenario maßgeblich die Leistungsfähigkeit der eingesetzten Verfahren. Für die simulative Bewertung wurden zwei verschiedene Szenarien ausgewählt, im Simulator nachgebildet und zur Simulation verwendet. Die Szenarien sind in Abschnitt 8.2 dokumentiert. Die Realisierung der Szenarien in der Simulationsumgebung, die Implementierung des Distanzschätzungsmodells sowie die Konfiguration des Simulators beschreibt Abschnitt 8.3.

Anschließend werden in Abschnitt 8.4 die Ergebnisse der Simulation vorgestellt und in Abschnitt 8.5 bewertend diskutiert. Abschnitt 8.6 rundet das Kapitel mit einer Zusammenfassung ab.

8.1 Modellbildung

Die Distanzschätzung auf Basis der Signalstärke beruht auf der Tatsache, dass sich die Funksignale, die ein Gerät aussendet, mit zunehmender Distanz abschwächen. Die meisten Funkinterfaces stellen zu diesem Zweck einen numerischen Wert zur Verfügung, der die Signalstärke der empfangenen Funkdaten und somit indirekt die Verbindungsqualität widerspiegelt. Dieser Wert wird daher im Allgemeinen *Received Signal Strength Indicator* (RSSI) oder *Link Quality Indicator* (LQI) genannt. Im Folgenden wird der zweite der beiden Begriffe stellvertretend für alle derartigen Verfahren verwendet. Allgemein lässt sich feststellen, dass Distanzschätzungen für kurze Distanzen recht genaue Werte liefern, für größere Distanzen sind die Ergebnisse jedoch oft sehr ungenau [103].

Im Rahmen dieser Evaluation stützen wir unsere Modellbildung auf Messwerte die im Freifeld aufgenommen wurden. Zur Messerhebung wurden iSense Sensorknoten (auf Basis des JN5139 Chipsatzes) verwendet und es wurde der LQI Wert bei verschiedenen Distanzen aufgenommen. Die Sensorknoten waren in Anlehnung an Messungen aus dem BSI-Projekt FleGSens auf einer Höhe von 20cm über dem Grund positioniert. Gemessen wurde alle 2m von 2m bis 42m. Bei jeder dieser Messungen haben zwei Knoten jeweils 480 Nachrichten gesendet (30 Nachrichten pro Kanal für die Kanäle 11 bis 26). Das entspricht $2 * 480 = 960$ Nachrichten pro Messung. Die Ergebnisse dieser Messung werden in Abschnitt 8.1.1 als Tabelle $T_{Messung}$ bezeichnet und sind in Abbildung 8.1 grafisch dargestellt. Zu sehen ist der Mittelwert des gemessen LQI über die Distanz aufgetragen sowie der Wert der Standardabweichung der Messungen, der in Form von vertikalen Linien aufgetragen ist. Der starke anfängliche Signalabfall ist deutlich zu erkennen und mit zunehmender Distanz flacht die Kurve sichtbar ab und die Standardabweichung nimmt zu.

Ein Single-hop-Distanzmesser (bzw. -schätzer) erhält als Eingabe den LQI Wert eines empfangenen Pakets und muss daraus eine Distanzschätzung ableiten. Er benötigt dazu die Distanz als Funktion des LQI und damit die Umkehrfunktion des hier dargestellten Verlaufs. Die in Abbildung 8.1 gezeigte Messreihe weist jedoch keine streng monoton fallenden Werte auf und ist somit nicht ohne weitere Verarbeitung invertierbar. In Abbildung 8.2 ist eine solche, manuell vorgenommene, Bearbeitung dargestellt. Die Abbildung zeigt die Mittelwerte aus Abbildung 8.1 und eine Kurve mit manuell adaptierten Werten, um zur Modellbildung eine streng monoton fallende Kurve zu erhalten. Im Folgenden werden diese adaptierten Mittelwerte des LQI zur weiteren Modellbildung verwendet. In Abschnitt 8.1.1 werden diese als Tabelle $T_{Monoton}$ bezeichnet und zur Entfernungsschätzung herangezogen.

Diese resultierende invertierbare Kurve kann in platzsparender Tabellenform auf Sensorknoten verwendet werden, um bei Empfang eines Paketes aus dessen LQI Wert eine Distanzschätzung abzuleiten. Innerhalb einer Simulationsumgebung ist die Fragestellung jedoch eine andere. Hier kennt der Simulator naturgemäß die exakten Positionen der einzelnen Sensorknoten und deren Distanzen.

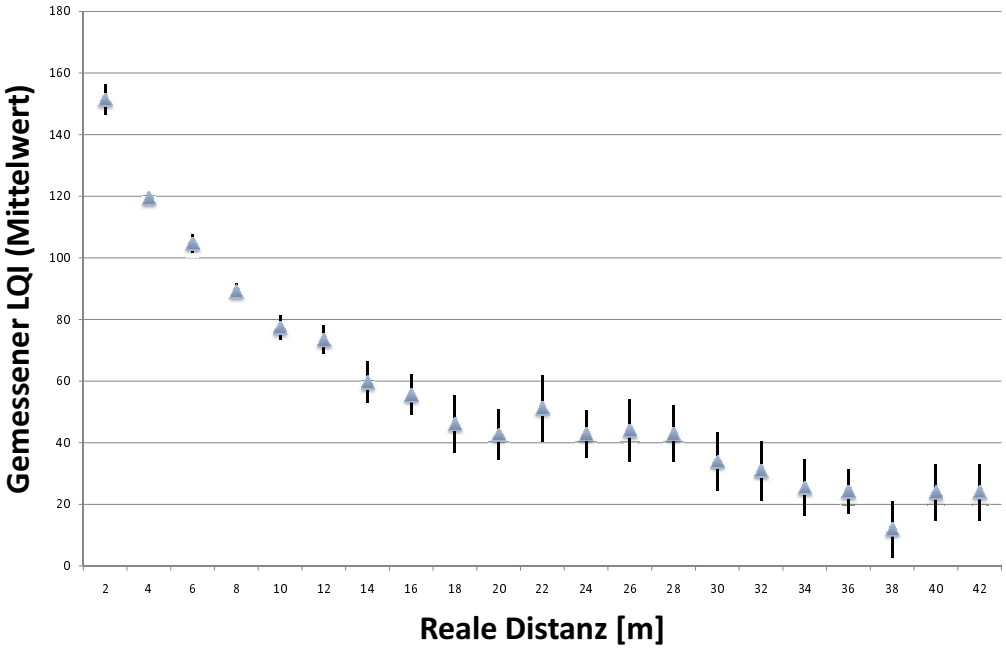


Abbildung 8.1: Gemessener LQI über die Distanz aufgetragen. Dargestellt ist der Mittelwert. Die vertikalen Linien geben die Standardabweichung an.

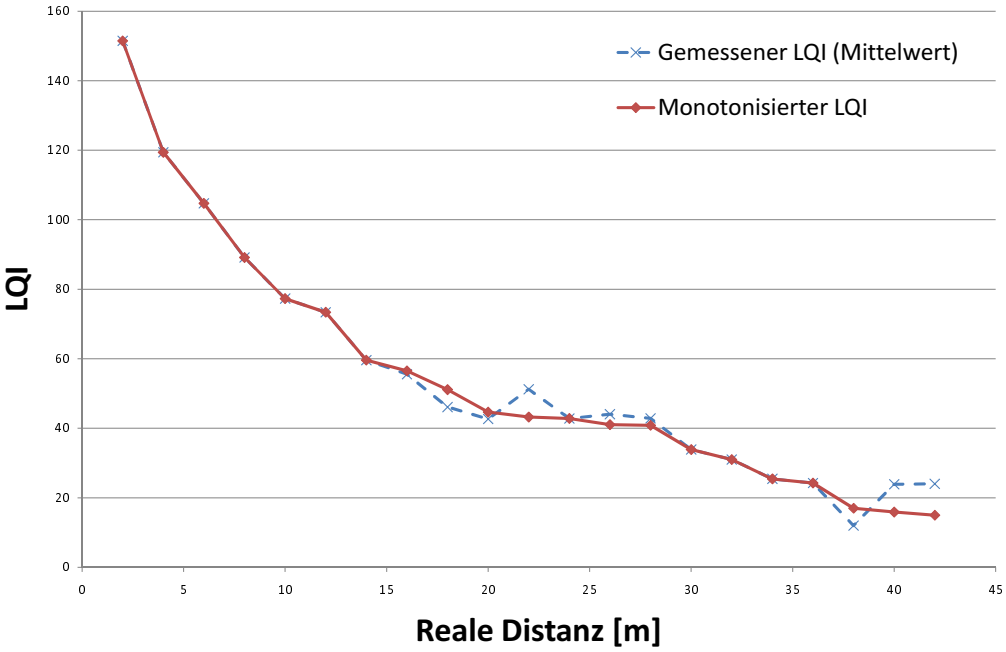


Abbildung 8.2: Mittelwerte des gemessenen LQI und manuell adaptierte Werte, um zur Modellbildung eine streng monoton fallende Kurve zu erhalten.

Die Schwierigkeit besteht nun darin, eine "unscharfe" Positionsschätzung des Sensorknoten zu generieren.

Eine mögliche Vorgehensweise ist es, die gemessenen Werte (Traces) direkt und ohne weitere Verarbeitung als Eingabe für einen Distanzschätzer in den Simulator zu speisen. Der Nachteil dieser sog. trace-basierten Variante ist, dass in der Implementierung eines solchen Modells im Simulator eine hardwarespezifische Verarbeitung vorgenommen wird. Dies liegt daran, dass direkt LQI-Werte, deren Zahlenwerte eine plattformabhängige Bedeutung haben, verwendet werden. Im Folgenden wird daher anstelle der Funktion $dist_{est} = f(lqi)$ einer generelle Funktion $dist_{est} = f(dist_{real})$ hergeleitet. Diese Vorverarbeitung kann prinzipiell auch mit auf anderen Hardwareplattformen aufgenommenen Messwerten durchgeführt werden und die Implementierung des Modells ist somit vielfältiger anwendbar. Die Funktion $dist_{est} = f(dist_{real})$ lässt sich aus den monotonisierten LQI Mittelwerten und den korrespondierenden Standardabweichungen numerisch aus den in Abbildung 8.1 und Abbildung 8.2 dargestellten Daten ermitteln. Diese enthalten Daten für Distanzwerte zwischen d_{min} und d_{max} (im Fall der oben dargestellten Messwerte ist $d_{min} = 2m$ und $d_{max} = 42m$). Die Transformation des hardwareabhängigen Modells hin zu einem hardwareunabhängigen Modell wird im folgenden Abschnitt schrittweise beschrieben.

8.1.1 Ermittlung der Genauigkeit

Als Eingabe für den in diesem Abschnitt beschriebenen Algorithmus dient eine Distanz d_1 zwischen d_{min} und d_{max}. Dies entspricht der tatsächlichen Distanz zwischen Sender und Empfänger, für welche zunächst ein LQI berrechnet wird. Anschließend wird aus diesem LQI-Wert im zweiten Schritt, welcher auf den Messungen aus $T_{Messung}$ basiert, eine Entfernungsschätzung ermittelt. Hierfür wird die im vorherigen Abschnitt beschriebene monotonisierte Kurve bzw. Tabelle $T_{Monoton}$ verwendet. Die Schritte lauten im Einzelnen:

1. Ermittle einen LQI für die gegebene Distanz d_1:
   - Ermittle anhand der in Tabelle $T_{Messung}$ enthaltenen Entfernungen die nächsthöhere (d_{high}) und nächstniedrigere (d_{low}) Entfernung für d_1.
   - Ermittle nun einen möglichen LQI Wert (lqi) für d_{high} (lqi_{high}) und d_{low} (lqi_{low}).
   - Verwende den zugehörigen Mittelwert ($mean_{high}$ und $mean_{low}$) und die Standardabweichung ($stddev_{high}$ und $stddev_{low}$) aus der Tabelle $T_{Messung}$ sowie eine Gaussche Normalverteilung ($gauss()$).
   - Berechne $lqi_{high} = gauss() * stddev_{high} + mean_{high}$ und $lqi_{low} = gauss() * stddev_{low} + mean_{low}$.
   - Berechne den Mittelwert beider LQI Schätzungen $lqi = \frac{lqi_{high}+lqi_{low}}{2}$.
2. Ermittle eine Distanzschätzung $dist_{est}$ aus lqi:

- Ermittle hierfür den nächsthöheren ($\widetilde{lqi}_{high}$) und nächstniedrigeren ($\widetilde{lqi}_{low}$) Tabelleneintrag aus Tabelle $T_{Monoton}$.
- Bestimme daraus $\widetilde{d}_{high}$ und $\widetilde{d}_{low}$ und berechne $dist_{est} = \frac{\widetilde{d}_{high}+\widetilde{d}_{low}}{2}$.

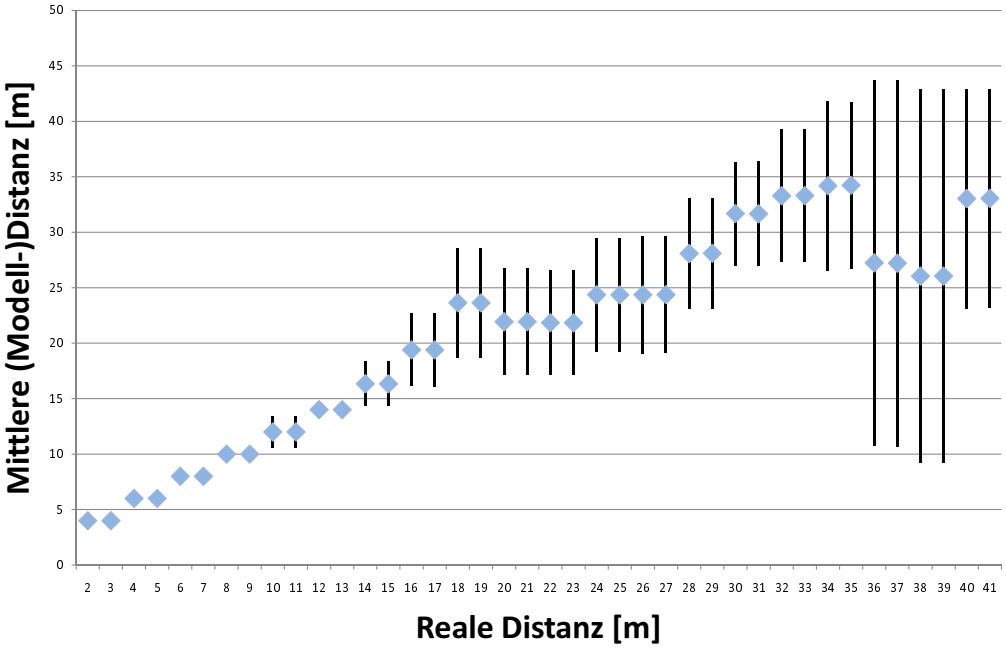


Abbildung 8.3: Mittelwerte und Standardabweichung der ermittelten Distanzschätzungen

Wiederholt man dieses Verfahren häufig, so erhält man für alle diskreten Eingabedistanzwerte d_i eine Menge von errechneten $dist_{est}$ Werten. Für jeden Eingabedistanzwert wird der Mittelwert und die Standardabweichung bestimmt und das Ergebnis in Tabellenform in einer Datei gespeichert. Für die durchgeführte simulative Evaluation wurden 1.000.000 zufällige Werte für d_1 gewählt. Das Ergebnis ist grafisch in Abbildung 8.3 und tabellarisch in Tabelle 8.1 dargestellt. Die geschätzte Distanz basiert folglich auf den monotonisierten Daten aus Tabelle $T_{Monoton}$, da der zugrundeliegende LQI jedoch auf den für die zugrundeliegende Entfernung gemessenen Daten aus Tabelle $T_{Messung}$ basiert, ist das in Abbildung 8.3 dargestellte Ergebnis nicht monoton.

8.2 Szenarien

In diesem Abschnitt werden die evaluierten Szenarien vorgestellt. Genau wie in der Realität beeinflusst die Wahl eines bestimmten Szenarios maßgeblich die erzielbaren Genauigkeiten von Lokalisationsalgorithmen. Aus diesem Grund ist es wenig sinnvoll, allgemeine Aussagen über ein Lokalisationsverfahren zu treffen. Die Güte der Lokalisierung muss immer im Kontext eines konkreten Szenarios

Reale Distanz [m]	Mittlere (Modell-)Distanz [m]	Standardabweichung
2,0	4,000	0,000
3,0	4,000	0,000
4,0	6,000	0,010
5,0	6,000	0,011
6,0	8,000	0,015
7,0	8,000	0,017
8,0	10,000	0,225
9,0	10,001	0,227
10,0	11,998	1,443
11,0	12,004	1,444
12,0	14,006	0,701
13,0	14,006	0,703
14,0	16,344	2,011
15,0	16,339	2,014
16,0	19,399	3,280
17,0	19,401	3,292
18,0	23,645	4,941
19,0	23,637	4,939
20,0	21,935	4,791
21,0	21,937	4,794
22,0	21,849	4,708
23,0	21,841	4,715
24,0	24,370	5,104
25,0	24,362	5,100
26,0	24,367	5,290
27,0	24,378	5,274
28,0	28,094	4,968
29,0	28,109	4,958
30,0	31,689	4,664
31,0	31,670	4,696
32,0	33,301	5,994
33,0	33,309	5,990
34,0	34,200	7,622
35,0	34,246	7,516
36,0	27,248	16,478
37,0	27,216	16,493
38,0	26,048	16,829
39,0	26,052	16,833
40,0	33,043	9,883
41,0	33,064	9,847

Tabelle 8.1: Mittelwerte und Standardabweichung der ermittelten Distanzschätzungen

bewertet werden. Im Rahmen dieser Evaluation wurden zwei verschiedene Szenarien betrachtet. Das erste der Szenarien ist ein Vergleichsszenario, das den Aufbau aus der bekannten Publikation von Langendoen et al. [97] nachbildet. Das zweite Szenario ist an das WSNLAB Projekt angelehnt und antizipiert mögliche Werte des Outdoor-Szenarios an der Universität Bonn.

Beide Szenarien erstrecken sich über eine Fläche der Größe $100m * 100m$, auf welcher jeweils n Sensorknoten zufällig verteilt werden. Eine Übersicht der verwendeten Parameter findet sich in Tabelle 8.2 und wird im Folgenden für die einzelnen Szenarien beschrieben. Ebenfalls wird für beiden Szenarien jeweils stellvertretend eine Grafik abgebildet. Diese stellen nur eine der 1000 verschiedenen Konfigurationen in Bezug auf die Knotenpositionen dar. Die anschlie-

ßende Evaluation betrachtet daher für jedes Szenario Ergebnisse die aus 1000 Simulationen mit verschiedenen Knotenpositionen errechnet wurden. Verantwortlich dafür ist die Wahl verschiedener sogenannter Seeds, also Zahlen die als Startwert für einen Pseudozufallszahlengenerator dienen.

Die weiteren Parameter des ersten Szenarios entsprechen den im *Standard Szenario* in [97] gewählten Parametern: Das in Abbildung 8.4 dargestellte Szenario 1 beinhaltet insgesamt 225 gleichverteilte Sensorknoten (Kreise), wovon 11 Anchorknoten darstellen (rot), von denen 9 Knoten anhand eines $3x3$ Gitters angeordnet sind und die restlichen 2 Anchor zufällig verteilt wurden. Die anhand des $3x3$ Gitters angeordneten Anchorknoten befinden sich folglich an den Positionen (25/25), (25/50), (25/75), (50/25), (50/50), (50/75), (75/25), (75/50) und (75/75), wodurch eine gleichmäßige räumliche Verteilung der Anchorknoten im Sensornetz erreicht wird. Die Kommunikationsreichweite der Sensorknoten in diesem Szenario beträgt 14m und Nachrichten werden im Netzwerk mit einem Flood Limit von 4 weitergeleitet. Der Wert für das Flood Limit bestimmt, wie viele Hops Nachrichten in einem Multi-Hop Netzwerk weitergeleitet werden. Der verwendete Wert 4 stammt aus dem Szenario aus [97].

Für die Simulation wurde das in Shawn zur Verfügung stehende CSMA Transmission Model mit den Parametern von IEEE 802.15.4 verwendet, wobei wie in [97] das CSMA Model lediglich für die Erzeugung von Delay verwendet wurde und nicht zur Simulation von Nachrichtenverlust. Hierfür wurde die Begrenzung maximalen Sendeversuche pro Nachricht entfernt, so dass alle gesendeten Nachrichten auch ausgeliefert (empfangen) werden. Als Distanzschätzer wurde der in [97] verwendete Relative Distanzschätzer mit einer Ungenauigkeit von 1,4m (10% der Kommunikationsreichweite) verwendet.

Das zweite Szenario unterscheidet sich vom ersten Szenario, wie in Abbildung 8.5 dargestellt, durch die Anzahl der im Sensornetz verteilten Sensorknoten und durch den Distanzschätzer. Szenario 2 beinhaltet insgesamt 50 gleichverteilte Sensorknoten (Kreise), wovon wieder 11 Anchorknoten darstellen (rot), welche derselben Anordnung wie in Szenario 1 folgen. Als Distanzschätzer wurde hier jedoch der in Abschnitt 8.1 vorgestellte Distanzschätzer verwendet. Die Sendereichweite wurde basierend auf den Ergebnissen der Modellentwicklung aus Abschnitt 8.1 auf 40m gesetzt. Diese Reichweite liegt weit unter der maximal möglichen Kommunikationsreichweite die in der Größenordnung von 100m liegt. Jedoch ist die Aussagekraft der LQI-Werte für größere Entfernungen vernachlässigbar und erlauben insbesondere keine sinnvollen Rückschlüsse auf die Distanz. Nachrichten werden im Netzwerk mit einem Flood Limit von 4 weitergeleitet.

8.3 Simulationsumgebung

Dieser Abschnitt beschreibt die verwendete Simulationsumgebung Shawn in Abschnitt 8.3.1. In Abschnitt 8.3.2 wird die Verwendung existierender Distanzschätzungsmodelle in Shawn eingeführt und im Anschluss wird das neu

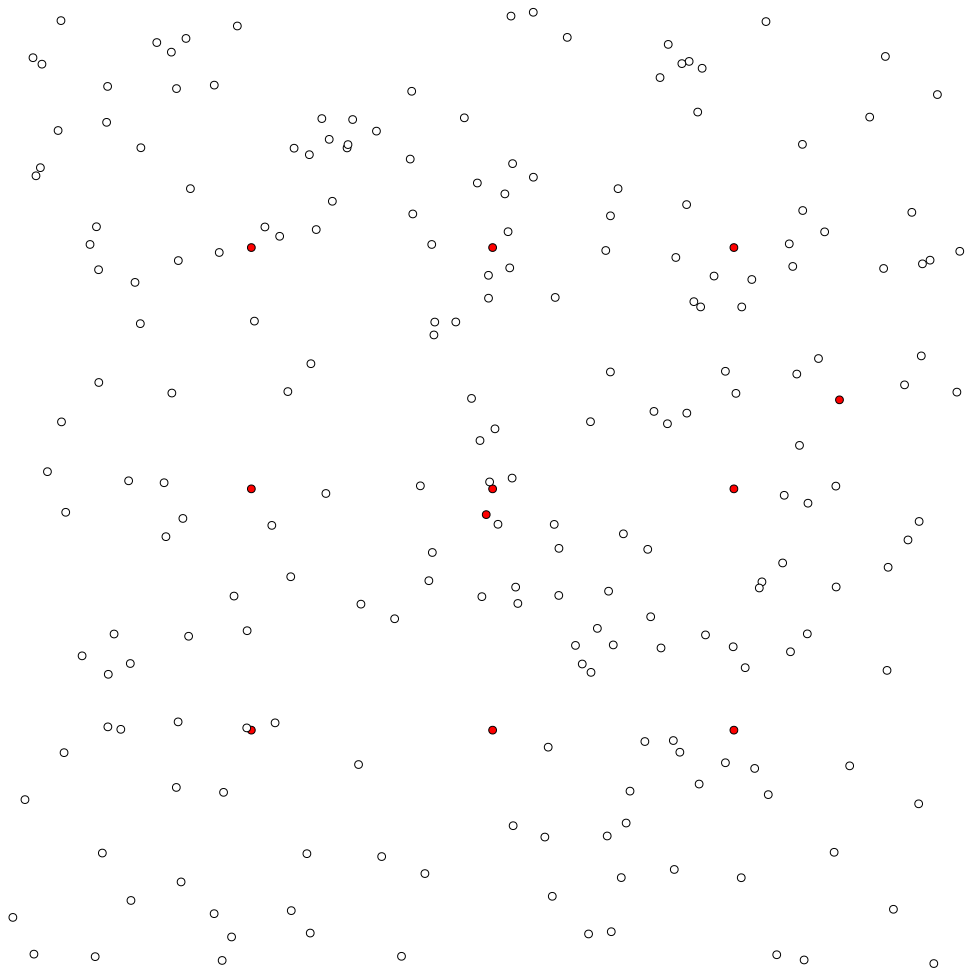


Abbildung 8.4: Visualisierung einer der 1000 Varianten von Szenario 1 (225 Knoten)

entwickelte LQI-basierte Distanzschätzungsmodell (vgl. Abschnitt 8.1) und dessen Verwendung dokumentiert.

8.3.1 Shawn

Die Evaluation der Lokalisierungsalgorithmen erfolgt in Shawn [56], einem Simulator für drahtlose Sensornetze. Im Gegensatz zu herkömmlichen Simulatoren für Sensornetzwerke, wie zum Beispiel Ns-2 [162] und TOSSIM [100], unterscheidet er sich in seiner Zielsetzung und Realisierung signifikant von eben diesen. So konzentriert sich Shawn darauf, die Auswirkungen von Phänomenen zu simulieren, statt die Phänomene selbst zu simulieren. Anstatt z.B. ein komplettes MAC-Protokoll (inkl. des Radiopropagationsmodells) zu simulieren, modelliert Shawn seine Effekte (z.B. Verlust, Verfälschung und Verzögerung). Simulationen werden dadurch aufgrund der im Detail bekannten Modellierung

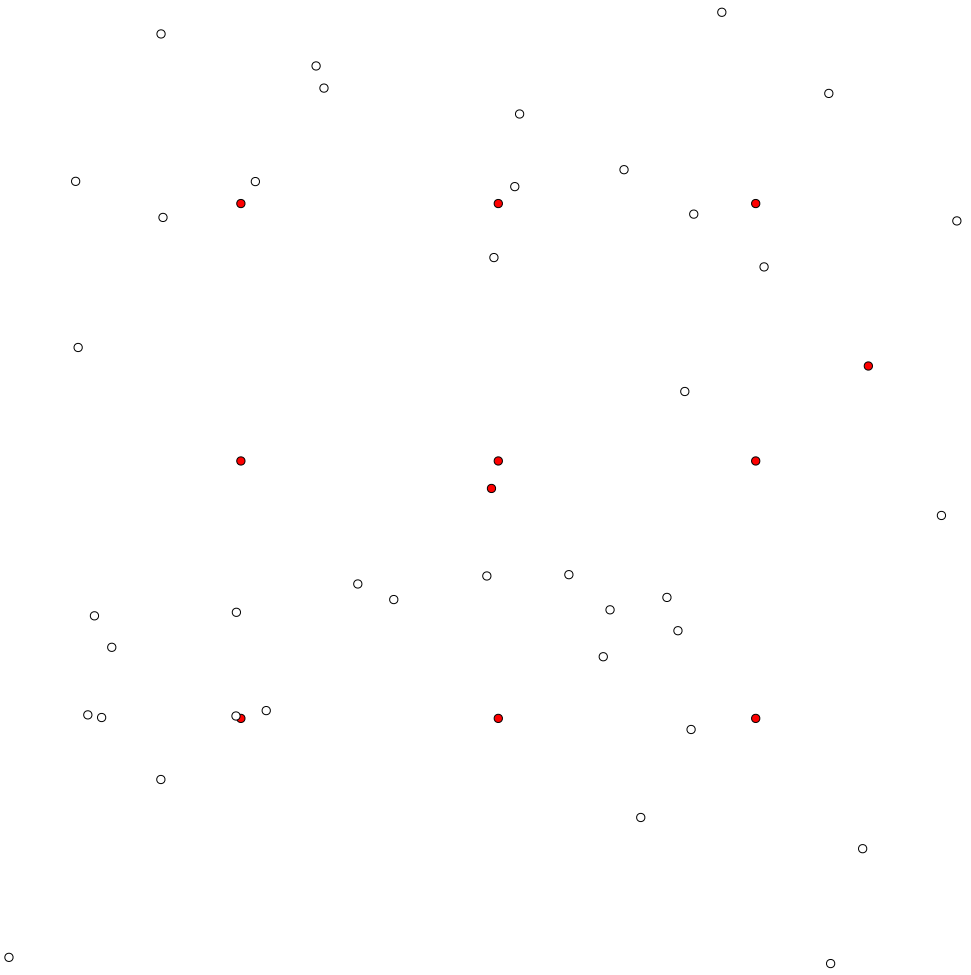


Abbildung 8.5: Visualisierung einer der 1000 Varianten von Szenario 2 (50 Knoten)

wiederholbarer und aussagekräftiger. Oft können diese Modelle extrem effizient implementiert werden, wodurch die Simulationszeit erheblich reduziert wird und auch große Knotenzahlen handhabbar werden. Mit der aktuellen Version von Shawn können auf Standard PC Komponenten Netzwerke mit weit mehr als 500.000 Knoten simuliert werden. Des Weiteren existiert eine Erweiterung von Shawn, so dass Anwendungen für die iSense-Hardwareplattform direkt ohne Änderungen im Anwendungscode auch auf Standard PCs simuliert werden können. Weitere Informationen zu Shawn finden sich in [56,93,129] und Details zu der Adaptionsschicht von iSense an Shawn finden sich in [128].

Da in WSNLAB zum Einen die simulative Evaluation eine wichtige Rolle spielt und zum Anderen auch verschiedene Hardwareplattformen zum Einsatz kommen, bietet die Wiselib (vgl. Abschnitt 3.3.3 und [16]) eine skalierbare und flexible Lösung, um dieselbe Implementierung auf allen Plattform einzusetzen. Daher werden in WSNLAB die Lokalisationsalgorithmen auf Basis der Wise-

Parameter	**Szenario 1**	**Szenario 2**
Fläche	100m×100m	
Anzahl Sensorknoten	225	50
Anzahl Anchorknoten	11	
Flooding Limit	4	
Kommunikationreichweite	14	40
Übertragungsmodell	Zuverlässig	
Distanzeschätzer	10% der Reichweite	LQI Distanzschätzer

Tabelle 8.2: Parameterübersicht der zwei Szenarien

lib zunächst im Simulator, der bereits optimal unterstützt wird, evaluiert. Auf diese Art und Weise werden doppelte Implementierungen vermieden und eine Fehlerbehebung kann an zentraler Stelle für alle Plattformen erfolgen. Ein weiterer Vorteil ist, dass jede neue von Wiselib unterstützte Plattform die implementierten Algorithmen verwenden kann. Es ist jedoch trotz prinzipieller Plattformunabhängigkeit davon auszugehen, dass bei der Portierung auf reale Hardwareplattformen noch eine Reihe von Problemen und Fehlern behoben werden müssen, die im Simulator nicht auftreten. Für den nächsten Meilenstein werden daher die entstandenen Implementierungen optimiert, getestet und die Adaption der Wiselib-Klassen an die in WSNLAB eingesetzten Hardwareplattformen vorangetrieben.

8.3.2 Distanzschätzungsmodelle

In Shawn sind sogenannte *distance estimates* für die Distanzschätzung zwischen zwei Knoten verantwortlich. Alle Distanzschätzer, die das C++-Interface *NodeDistanceEstimate* implementieren, können dann durch Parameter in einer Konfigurationsdatei erzeugt und von der Simulation verwendet werden. Ein solches Modell muss die Schnittstelle *NodeDistanceEstimate* implementieren, die hier verkürzt zu sehen ist:

```
1  class NodeDistanceEstimate
2  {
3      ...
4      virtual bool estimate_distance(
5              const Node& source,
6              const Node& target,
7              double& result
8          ) const throw() = 0;
9      ...
10 }
```

Die Methode liefert als Ergebnis, ob eine Distanzschätzung zwischen den zwei Knoten *source* und *target* ermittelt werden konnte. Dies kann modellabhängig scheitern, z.B. wenn die Entfernung zwischen den beiden Knoten größer als der Kommunikationsradius ist. Konnte eine Entfernung ermittelt werden, so wird das Ergebnis in der Variable *result* zurückgeliefert.

Aktuell enthält Shawn bereits drei verschiedene Distanzschätzungsalgorithmen. Im Folgenden werden die drei bereits von Shawn mitgelieferten Implementierungen und deren Funktionsweise kurz beschrieben. Zur Beschreibung wird ergänzend eine objektorientierte Pseudocodenotation verwendet. Die Funktion $real\_position()$ liefert die dem Simulator bekannte reale Position zurück und $euclidean\_norm()$ errechnet die euklidische Distanz. Die dargestellten Pseudocodefragmente reflektieren die Implementierung im Simulator und basieren auf der Beschreibung in [127].

PerfectDistanceEstimate: Liefert eine perfekte, 100% korrekte Distanzschätzung zwischen zwei Knoten *source* und *target* erfolgt nach folgender Vorschrift: $result = (source.real\_position() - target.real\_position()).euclidean\_norm()$. Dieser Distanzschätzer ist insbesondere hilfreich um die Güte eines Lokalisierungsverfahrens zu evaluieren ohne den Einfluss externen Störgrößen, wie sie beispielsweise durch Messfehler verursacht werden.

AbsoluteErrorDistanceEstimate: Addiert einen absoluten Fehler (bestehend aus konstantem Offset und zufälligem absoluten Fehler) auf die realen Distanzen. Er erwartet die drei Parameter *name*, *offset* und *error*. Der Name ist beliebig und dient der späteren Referenzierung in der Konfiguration des Simulators. Die Berechnung einer Distanz zwischen zwei Knoten *source* und *target* erfolgt nach folgender Vorschrift: $result = offset + uniform\_random(-\frac{error}{2}, +\frac{error}{2}) + (source.real\_position() - target.real\_position()).euclidean\_norm()$.

RandomizedDistanceEstimate: Dieser Distanzschätzer fügt der korrekten Distanzschätzung eine zufällige additive und multiplikative Komponente hinzu. Er erwartet die (optionalen) Parameter *name* (Standard: leer), *multiplier* (Standard: null), *offset* (Standard: null), *chop_low* (Standard: `std::numeric_limits<double>::min()`) und *resample_chopped* (Standard: false).

Zur Ermittlung der Schätzung wird ein Distanzschätzer mit zwei (optionalen) Zufallsvariablen *multiplier* und $offset$ konfiguriert. Zu Beginn einer Distanzschätzung zwischen zwei Knoten *source* und *target* wird zunächst $result = 0.0$ gesetzt. Ist ein Wert für *multiplier* konfiguriert, so berechnet dieser Distanzschätzer $result+ = source.real\_position() - target.real\_position()).euclidean\_norm() * multiplier()$ wobei $multiplier()$ die nächste Zufallszahl von der Zufallsvariablen *multiplier* abruft. Ist ein Wert für $offset$ konfiguriert, so berechnet dieser Distanzschätzer $result+ = offset()$ wobei $offset()$ die nächste Zufallszahl von der Zufallsvariablen $offset$ abruft.

Optionalerweise kann der Wertebereich der Distanzschätzungen nach unten limitiert werden, um zu kleine Werte (z.B. bei einer Gauss'chen Verteilung) zu vermeiden, die besonders bei der multiplikativen Komponente das Gesamtergebniss verzerren können. Dazu kann der Parameter *chop_low* gesetzt werden. Ist dieser gesetzt, so überprüft der Distanzschätzer, ob

der errechnete Wert kleiner als *chop_low* ist und setzt ggf. das Ergebnis auf den Wert von *chop_low*. Ist des weiteren *resample_chopped* auf *true* gesetzt, so werden bis zu 50.000 Zufallszahlen gezogen bis ein Wert größer als *chop_low* errechnet wurde. Ist auch nach 50.000 Ziehungen das Ergebnis noch kleiner als *chop_low*, so wird *chop_low* als Ergebnis zurückgegeben.

In Shawn können für die Simulationen beliebige Distanzschätzer verwendet werden. Für die simulative Evaluation in WSNLAB wurde unter anderem auf das bestehende *PerfectDistanceEstimate* zurückgegriffen um die Güte eines Verfahrens ohne Störeinflüsse bewerten zu können. Des Weiteren wurde ein neues Modell namens *LQI-based Distance Estimate* implementiert, das das Modell aus Abschnitt 8.1 implementiert. Das Modell erwartet eine Konfigurationsdatei, die jeweils für eine bestimmte, reale Distanz einen Mittelwert und eine Standardabweichung angibt. Abbildung 8.6 zeigt einen Auszug einer solchen Konfigurationsdatei, die die Werte aus Tabelle 8.1 enthält.

```
2.0  4.0  0.0
3.0  4.0  0.0
4.0  6.000023953912672  0.011988472631368078
5.0  6.0  0.01265078065394799
6.0  7.999992001983508  0.019997558846969396
7.0  8.000008004931038  0.015496736050951658
8.0  9.999447876324297  0.23130043926356209
9.0  9.999593359804813  0.2284589848858717
10.0  11.996460970590826  1.4441790566587407
```

Abbildung 8.6: Auszug aus der Konfigurationsdatei für das LQI-basierte Distanzschätzungsmodell in Shawn (Spalten: Distanz in Meter, Mittelwert, Standardabweichung)

Zur Erstellung eines neuen Distanzschätzers, der diese Datei als Eingabe verwendet, muss in der Konfigurationsdatei der Simulation ein neuer LQI-basierter Distanzschätzer mit den Parametern *name*, *filename* und *max_range* erstellt werden. Der Parameter *max_range* gibt die maximale (reale) Entfernung von zwei Knoten an für die noch eine Distanzschätzung ermittelt werden soll. Ist die reale Entfernung größer, so liefert `bool estimate_distance(. . .)` false zurück. Die folgende Zeile ist ein Beispiel für die Erzeugung einer neuen Instanz mit dem Namen "lqi_err", die die Daten aus der Datei "outdoor.lqi" liest und als maximale Entfernung 50 verwendet.

```
create_lqi_distance_estimate name=lqi_err filename=outdoor.lqi
    max_range=50
```

Die Daten aus der Datei enthalten Werte zwischen den Distanzen d_{min} und d_{max}. Für Distanzen d mit $d_{min} \leq d \leq d_{max}$ ermittelt der LQI-basierter Distanzschätzer zunächst die beiden Einträge zwischen denen d liegt (im Sonderfall, dass genau ein Eintrag gefunden wird, wird dieser verwendet). Für diese(n) wird (jeweils) eine Distanzschätzung mit dem (jeweiligen) Mittelwert und der Standardabweichung errechnet und ggf. der Mittelwert zwischen beiden gebildet. Für Werte von d mit $0 \leq d < d_{min}$ wird die reale Distanz als Mittelwert und die Standardabweichung des Eintrags von d_{min} verwendet. Für Werte d mit

$d > d_{max}$ wird die reale Distanz als Mittelwert und die Standardabweichung des Eintrags von d_{max} verwendet.

Wird die Wiselib in Shawn verwendet, so muss der von der Wiselib zu verwendende Distanzschätzer konfiguriert werden. Dies geschieht über die Shawn-Variable *est_dist*, in der der Name des zu verwendenden Distanzschätzers enthalten sein muss. Dieser Name entspricht dem Parameter *name*, der einer neuen Distanzschätzerinstanz übergeben wurde oder der feste Name "perfect_estimate" im Falle des konfigurationsfreien Modells *PerfectDistanceEstimate*. Für das obige Beispiel würde das folgende Konfigurationsfragment die Verbindung zwischen der Wiselib und der konkreten Instanz des Distanzschätzers herstellen:

```
1| est_dist=lqi_err                                                    |
```

8.4 Ergebnisse

Im folgenden Abschnitt werden die Ergebnisse der Simulationen der Lokalisierungsverfahren vorgestellt. Untersucht wurde hierbei die Genauigkeit des Lokalisierungsergebnisses der in Kapitel 8 genannten Kombinationen von Lokalisierungsverfahren. Hierbei wurden die beiden Positionsbestimmungsverfahren Min-Max sowie Lateration mit den drei Multi-hop-Distanzschätzern Sum-Dist, Distance-Vector-Hop (DV-Hop) und Euclidean kombiniert, so dass die folgenden Verfahren untersucht wurden:

1. a) Sum-Dist mit Min-Max
   b) DV-Hop mit Min-Max
   c) Euclidean mit Min-Max
2. a) Sum-Dist mit Lateration
   b) DV-Hop mit Lateration
   c) Euclidean mit Lateration

Um die Genauigkeit der einzelnen Lokalisierungsverfahren vergleichen zu können, wurde jede der oben genannten Kombinationen in den im Abschnitt 8.2 beschriebenen Szenarien 1 und 2 mit den zugehörigen Parametern simuliert. Die Simulation wurde in jedem dieser Fälle 1000 mal wiederholt, wobei bei jeder Simulation ein unterschiedlicher Seed für die Simulation verwendet wurde. Die Wahl des Seeds beeinflusst alle Vorgänge einer Simulation bei der randomisierte Prozesse stattfinden. Beispiele sind die Platzierung der Knoten oder die Ziehung von Zufallszahlen zur Ermittlung des Backoffs bei CSMA/CA. Die Simulation eines Lokationsverfahrens endet mit der Bestimmung einer Position auf jedem Knoten. Es handelt sich also um terminierende Simulationen, bei denen keine anfängliche Einschwingzeiten beachtet werden müssen bis sich ein stabiles

Ergebnis einstellt. Daher wird bei der Evaluation auch keine Simulationszeit angegeben.

Am Ende jeder Simulation wurde für jeden Sensorknoten (mit Ausnahme der Anchor Knoten) die euklidische Distanz zwischen der tatsächlichen Position und der vom Sensorknoten ermittelten Position berechnet. Falls ein Sensorknoten seine Position nicht schätzen konnte, so wurde sein Positionsfehler nicht für den Vergleich der Genauigkeit der Lokalisierungsverfahren herangezogen, sondern getrennt als Prozentsatz der nicht-lokalisierten Sensorknoten angegeben. Für die Evaluation gilt ein Sensorknoten als nicht-lokalisiert, falls er seine Position nicht korrekt schätzen konnte, d.h. wenn er keine oder nicht ausreichend Nachrichten von den Anchorknoten erhalten konnte. Der erste Fall, in welchem ein Sensorknoten keine Nachrichten von den Anchorknoten erhalten konnte, entspricht eine Positionsschätzung auf den Ursprung des Koordinatensystems (0/0), während der zweite Fall einem Positionsfehler des Sensorknotens entspricht, welcher größer als die Szenariofläche ist. Zusammenfassend wurden folglich Positionsschätzungen als nicht-lokalisiert definiert:

1. Knoten mit geschätzte Position==(0/0) und tatsächlicher Position !=(0/0)
2. Knoten mit Positionsfehler $P_{err} \geqslant \sqrt{100^2 + 100^2} \approx 141$

Die Positionsfehler der verbleibenden Sensorknoten wurden getrennt nach Szenario und Lokalisierungsverfahren für alle 1000 Seeds hinsichtlich des Minimums, Medians, Maximums und der Quartilsabstände untersucht, was im Folgenden beschrieben wird.

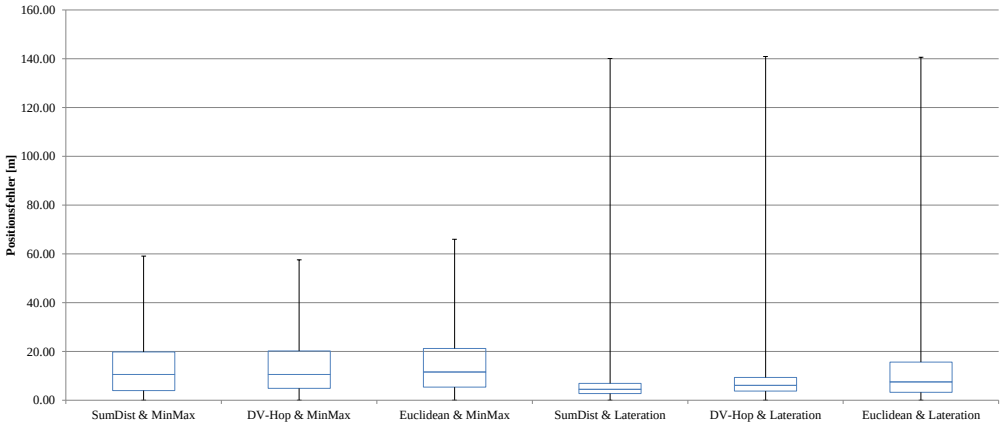
Abbildung 8.7(a) stellt das Ergebnis der einzelnen Lokalisierungsverfahren in Szenario 1 (225 Sensorknoten, 14m Kommunikationsreichweite) in Form eines Boxplots und in Tabelle 8.3 tabellarisch dar. Hierbei sind die Ergebnisse getrennt für die insgesamt 6 Lokalisierungsverfahren nebeneinander auf der X-Achse aufgetragen. Die Y-Achse zeigt den Positionsfehler in Metern. Dargestellt sind jeweils der minimale und maximale Positionsfehler in allen 1000 Simulationen (oberer und unterer Whisker), der Median (Linie in der Mitte der Box) sowie das untere und obere Quartil (die Box) für jedes Verfahren. Es ist zu erkennen, dass sich die Ergebnisse der drei auf Min-Max basierenden Verfahren sowie der drei auf Lateration basierenden Verfahren jeweils stark ähneln.

Bei den auf Min-Max basierenden Verfahren beträgt der Median etwa 11m, das Minimum etwa 0m und das Maximum ungefähr 60m, während die Quartilsabstände zwischen 4-5m und 19-21m betragen. Bei den auf Lateration basierenden Verfahren hingegen beträgt der Median etwa 6m, das Minimum etwa 0m und das Maximum ungefähr 140m, während die Quartilsabstände zwischen 3m und 7-9m (Euclidean: 16m) betragen.

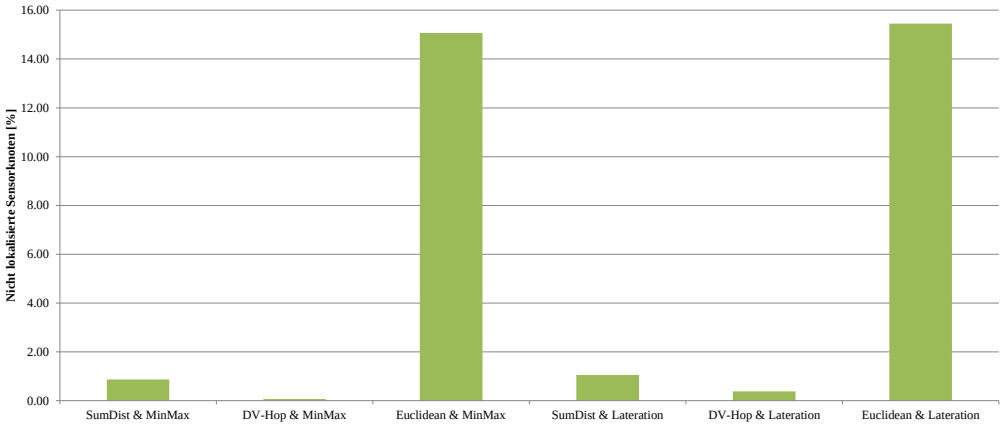
Anhand der Quartilsabstände ist zu erkennen, dass 50% der Sensorknoten in Szenario 1 bei den auf Min-Max basierenden Verfahren einen Positionsfehler von 4-21m aufweisen, während bei den auf Lateration basierenden Verfahren der Positionsfehler lediglich 3-9m beträgt, diese Verfahren bei der Hälfte der

Sensorknoten also doppelt so gute Ergebnisse liefern. Anhand der Maxima der einzelnen Lokalisierungsverfahren ist zu erkennen, dass die auf Lateration basierenden Verfahren deutlich höhere maximale Positionsfehler aufweisen.

Abbildung 8.7(b) stellt den Anteil der nicht-lokalisierten Sensorknoten dar, welcher für die auf dem SumDist und DV-Hop basierenden Verfahren bei ca. 1% liegt, während der Wert bei den auf Euclidean basierenden Verfahren bei deutlich höheren 15% liegt.



(a) Boxplot der Ergebnisse der Lokalisierung



(b) Vergleich der Lokalisierungsverfahren

Abbildung 8.7: Boxplot der Ergebnisse der Lokalisierung und Vergleich der Lokalisierungsverfahren in Szenario 1

Abbildung 8.8(a) stellt das Ergebnis der einzelnen Lokalisierungsverfahren in Szenario 2 (50 Sensorknoten, 40m Kommunikationsreichweite) in Form eines Boxplots und in Tabelle 8.4 tabellarisch dar. Die Darstellung der Ergebnisse erfolgt analog zu der Darstellung für Szenario 1 und 2.

Bei den auf Min-Max basierenden Verfahren beträgt der Median etwa 21m (Euclidean: 10m), das Minimum etwa $\leqslant$1m und das Maximum ungefähr 50-77m,

	SumDist MinMax	DV-Hop MinMax	Euclidean MinMax	SumDist Lateration	DV-Hop Lateration	Euclidean Lateration
Max	59.1	57.5	66.0	140.1	140.9	140.5
3. Quartil	19.8	20.2	21.2	6.9	9.4	15.7
Median	10.5	10.6	11.6	4.5	6.1	7.5
1. Quartil	4.0	4.9	5.4	2.8	3.7	3.3
Min	$\leqslant$0.1	$\leqslant$0.1	$\leqslant$0.1	$\leqslant$0.1	$\leqslant$0.1	$\leqslant$0.1

Tabelle 8.3: Wertetabelle der Ergebnisse der Lokalisierung in Szenario 1

während die Quartilsabstände zwischen 12-29m (Euclidean: 6-17m) betragen. Sowohl bei den auf MinMax wie bei den auf Lateration basierenden Verfahren ist eine Verbesserung des Lokalisierungsergebnisses von SumDist über DV-Hop zu Euclidean zu beobachten.

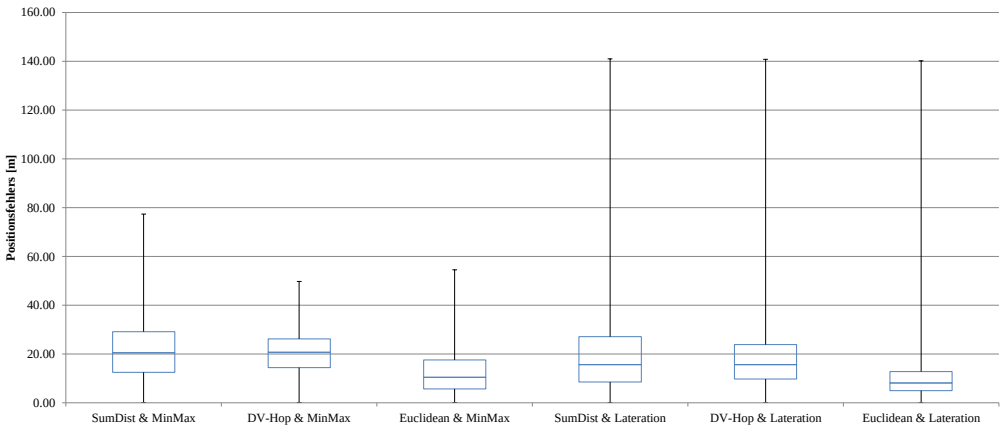
Abbildung 8.8(b) stellt den Anteil der nicht-lokalisierten Sensorknoten dar. Die Ergebnisse hinsichtlich der nicht-lokalisierten Sensorknoten gleichen in Szenario 2 den Ergebnissen aus Szenario 1, wobei die beiden Euclidean-basierten Verfahren, genau wie in Szenario 1, als einzige einen Anteil von mehr als 1% aufweisen. In Szenario 2 liegt dieser Anteil noch einmal deutlich höher (ca. 34%), was auf die geringere Netzdichte (225 Knoten gegenüber 50 auf identischer Fläche) zurückzuführen ist und nur teilweise durch die höhere Sendereichweite (40m gegenüber 14m) kompensiert wird.

	SumDist MinMax	DV-Hop MinMax	Euclidean MinMax	SumDist Lateration	DV-Hop Lateration	Euclidean Lateration
Max	77.3	49.8	54.5	140.9	140.7	140.1
3. Quartil	29.1	26.2	17.6	27.1	23.8	12.8
Median	20.6	20.7	10.5	15.6	15.6	8.2
1. Quartil	12.5	14.5	5.7	8.5	9.7	5.0
Min	$\leqslant$0.1	$\leqslant$0.1	$\leqslant$0.1	$\leqslant$0.1	$\leqslant$0.1	$\leqslant$0.1

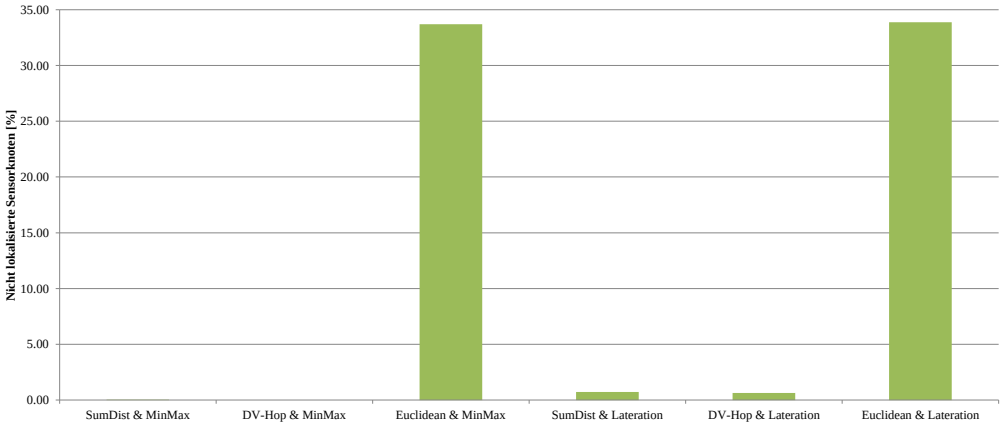
Tabelle 8.4: Wertetabelle der Ergebnisse der Lokalisierung in Szenario 2

8.5 Diskussion und Bewertung

Vergleicht man die in einzelnen Lokalisierungsverfahren in Szenario 1 anhand der in Abbildung 8.7(a) und Abbildung 8.7(b) dargestellten Daten, so ist zusammenfassend erkennbar, dass Lateration eine bessere durchschnittliche Genauigkeit als MinMax aufweist, jedoch auch deutlich größere maximale Positionsfehler hervorbringt. Darüber hinaus ist erkennbar, dass im Mittel SumDist bessere Ergebnisse als DV-Hop liefert und DV-Hop bessere Ergebnisse als Euclidean liefert und dass Euclidean mit großem Abstand den höchsten Anteil an nicht-lokalisierten Sensorknoten generiert (Faktor 15).



(a) Boxplot der Ergebnisse der Lokalisierung



(b) Vergleich der Lokalisierungsverfahren

Abbildung 8.8: Boxplot der Ergebnisse der Lokalisierung und Vergleich der Lokalisierungsverfahren in Szenario 2

Der um den Faktor 15 größere Anteil an nicht-lokalisierten Sensorknoten bei Euclidean im Gegensatz zu den anderen Verfahren liegt an der in [97] beschriebenen Verfahrensweise von Euclidean, um die Multihop Entfernung zu den Anchorn zu bestimmen. Damit ein Knoten seine Entfernung zu einem Anchor mittels Euclidean berechnen kann, muss er entweder vom Anchor selbst eine Nachricht empfangen haben oder aber von mindestens zwei anderen Knoten, die ihre Entfernung zum Anchor und zueinander kennen, eine Nachricht empfangen haben. Bei der Entfernungsberechnung wird anschließend das *neighbor vote* und das *common neighbor* Verfahren eingesetzt, um bei mehreren möglichen Entfernungen die wahrscheinlichste Entfernung auszuwählen. Erhält ein Knoten jedoch von weniger als zwei solcher Knoten eine Nachricht oder können die neighbor Verfahren kein Ergebnis liefern, so kann der Knoten seine Entfernung zum Anchor nicht berechnen und er leitet keine Nachricht an seine Nachbarn weiter. Bei der Verwendung von SumDist und DV-Hop hingegen kann immer

eine Entfernung berechnet werden und die Nachricht wird auch immer bis zum gewählten Flooding-Limit weitergeleitet.

In Szenario 1 empfiehlt sich folglich der Einsatz des SumDist Verfahrens in Kombination mit dem Lateration Verfahren, da dieses in diesem Szenario die besten Lokalisierungsergebnisse liefert.

Vergleicht man die in einzelnen Lokalisierungsverfahren in Szenario 2 anhand der in Abbildung 8.8(a) und Abbildung 8.8(b) dargestellten Daten, so ist zusammenfassend erkennbar, dass Lateration genau wie in Szenario 1 eine bessere durchschnittliche Genauigkeit als MinMax aufweist, jedoch auch größere Quartilsabstände sowie deutlich größere maximale Positionsfehler hervorbringt. Darüber hinaus ist erkennbar, im Gegensatz zu Szenario 1 SumDist hier schlechtere Ergebnisse als DV-Hop liefert und DV-Hop schlechtere Ergebnisse als Euclidean liefert. Dies liegt an verringerten Netzdichte in Szenario 2 im Vergleich zu Szenario 1 in Kombination mit der dadurch erhöhten Anteil an Anchorknoten. In Szenario 2 kann in diesem Fall keine eindeutige Empfehlung ausgesprochen werden, da hier ein Tradeoff zwischen durchschnittlicher Genauigkeit und Quartilsabständen gewählt werden muss. Im Zusammenhang mit dem anderen Szenario kann Euclidean jedoch bereits jetzt als am wenigsten geeigneter Kandidat für die Lokalisierung identifiziert werden, da hier der Anteil an nicht-lokalisierten Sensorknoten in beiden Szenarien deutlich höher ist als bei den verbleibenden Verfahren.

Der Vergleich der beiden Szenarien insgesamt zeigt – wie bereits zu Beginn von Kapitel 8 beschrieben, dass die Wahl des Lokalisierungsverfahrens und die Parameter des gewählten Szenarios angepasst werden müssen, damit das Lokalisierungsverfahren gute Ergebnisse liefern kann. Darüber hinaus zeigt der Vergleich von Szenario 1 und 2, dass das WSNLab Szenario mit 50 Sensorknoten und 11 Anchorknoten bei einer Kommunikationsreichweite von 40m prinzipiell Lokalisierungsergebnisse liefert. Normalisiert man den mittleren Fehler in Szenario 2 ($\leqslant$22) auf die Kommunikationsreichweite von 40m, so liefert der in Abschnitt 8.1 beschriebene Distanzschätzer eine Genauigkeit von 45% bis 55% der Kommunikationsreichweite.

8.6 Zusammenfassung

Durch die in diesem Bericht durchgeführten Simulationen konnte gezeigt werden, dass die implementierten Lokalisierungsverfahren Ergebnisse liefern, welche zu den in [97] beschriebenen passen. So lässt sich anhand der vorgestellten Ergebnisse reproduzieren, dass die Ergebnisse von Euclidean stark von der Konnektivität abhängen (hoher Anteil an nicht-lokalisierten Knoten), jedoch bei vorhandener Konnektivität die besten Ergebnisse liefert (vgl. vor allem Szenario 2).

Insgesamt lässt sich feststellen, dass sich die Multi-hop-Lokalisierung auf Basis von Signalstärkenmessungen nur bedingt zur Bestimmung präziser Positions-

informationen eignet. Dieses lässt sich direkt auf die Charakteristik der Signalstärkenmessungen zurückführen: Während ihr Verlauf für kleine Distanzen steil ist und somit recht zuverlässige Aussagen über die Distanz zwischen Sender und Empfänger zulässt, flacht der Verlauf mit zunehmender Distanz immer mehr ab. Somit wird es schwieriger, zuverlässig Rückschlüsse auf die tatsächliche Distanz zu ziehen. Die sich daraus ergebenden Ungenauigkeiten pflanzen sich in die Positionsbestimmung fort und führen zu teilweise erheblichen Fehlern.

Eine mögliche Vorgehensweise wäre es, lediglich solche Distanzschätzungen in die Positionsbestimmung einzubeziehen, die aus dem steilen Bereich der Signalstärkenkurve stammen. Dieser Weg wurde bereits ansatzweise eingeschlagen, indem in Szenario 2 eine Kommunikationsreichweite von 40m angenommen wurde, die deutlich unter den tatsächlichen Reichweite von ca. 100m liegt. Für eine weitere Verbesserung der Ergebnisse müsste die maximal schätzbare Distanz allerdings deutlich weiter gesenkt werden. Nachteilig an diesem Vorgehen ist jedoch, dass die abschätzbaren Distanzen zunehmend nur einen kleinen Teil der tatsächlichen Kommunkationsreichweite umfassen, und dass insgesamt weniger Distanzschätzungen zur Verfügung stehen.

Eine nachhaltige Verbesserung der Situation ließe sich lediglich mit einem Verfahren zur Single-hop-Distanzschätzung erreichen, dessen Genauigkeit nicht mit der Distanz abnimmt. Erfahrungen mit ähnlichen Technologien deuten an, dass die Messung der Signalumlaufzeit ein solches Verfahren sein könnte. Es bleibt abzuwarten, inwieweit sich diese Hoffnungen im Rahmen von Referenzmessungen im weiteren Projektverlauf bestätigen lassen.

9 Lokalisierungsanwendung

Dieser Abschnitt beschreibt die Implementierung der Lokalisierungsanwendung und damit ebenfalls der Lokalisierungsverfahren, die Kapitel 7 beschrieben und Kapitel 8 simulativ evaluiert wurden. In diesem Kapitel wird die Anwendung der verschiedenen Lokalisierungsverfahren auf realer Hardware untersucht und diskutiert welche Probleme bei der Transition von der Simulation zu echter Hardware aufgetreten sind. Des Weiteren werden Messergebnisse vorgestellt.

Im Rahmen der Arbeiten für diesen Bericht wurden die in den vorherigen Berichten genannten Verfahren auf realer Hardware implementiert, kompiliert und getestet. Die verwendeten Hardwareplattformen sind iSense und TelosB und die verwendbaren Betriebssysteme sind iSense, Contiki und TinyOS. Die implementierten Verfahren sind die folgenden:

- Single-Hop Distanzschätzung
  - LQI-basierte Distanzschätzung
  - Time-of-Flight-basierte Distanzschätzung
- Multi-Hop Distanzschätzung
  - Sum-Dist
  - DV-Hop
  - Euclidean
- Positionsberechnung
  - Min-Max
  - Lateration

Alle Implementierungen basieren auf der Wiselib und sind somit prinzipiell auf allen von ihr unterstützten Plattformen direkt einsetzbar. Die einzelnen Verfahren sind als zueinander kompatible Module implementiert, die in beliebiger Kombination eingesetzt werden können. Es ist damit auf einer gemeinsamen Codebasis möglich, Lokalisierungsverfahren und die Lokalisierungsanwendung sowohl im Simulator als auch auf realer Hardware zu evaluieren. Zur Erlangung der hier vorgestellten Ergebnisse wurde die iSense Plattform verwendet, da diese auch über die Hardware zur Messung der Herzfrequenz verfügt und im Rahmen des Outdoor-Tests eingesetzt werden soll.

Die Verfahren Lateration und Min-Max wurden in den vorgenannten Berichten als besonders geeignet identifiziert und werden daher auch hier evaluiert. Zur Positionsbestimmung einzelner Knoten müssen diese die Multi-Hop Distanzen

zu den Ankerknoten ermitteln. Um die Multi-Hop Distanzen aus den jeweiligen Single-Hop-Distanzschätzungen zu ermitteln, wurden die Verfahren Sum-Dist, DV-Hop und Euclidean vorgestellt. Aufgrund des Ressourcenbedarfs von Euclidean und aufgrund des vergleichsweise schlechten Lokalisierungsanteils wurde dieses Verfahren nicht weiter betrachtet. Ein weiteres Problem bei Euclidean ist, dass Nachrichten zwischen den Knoten ausgetauscht werden müssen, die größer als die Maximum Transfer Unit (MTU) von IEEE 802.15.4 (127 Byte) sind. Es müssten daher noch zusätzliche Vorkehrungen zur (De-) Fragmentierung von Paketen eingesetzt werden, die zusätzlich Ressourcen verbrauchen. Zur Ermittlung der Distanzschätzungen kam das LQI-basierte Verfahren zum Einsatz (vgl. Abschnitt 8.1). Die Time-of-Flight-basierte Distanzschätzung wurde zunächst getrennt evaluiert (siehe Abschnitt 10.2).

Im Folgenden wird zunächst die Architektur der Anwendung vorgestellt (vgl. Abschnitt 9.1). Im Anschluss werden in Abschnitt 9.2 die beiden Testumgebungen vorgestellt, in denen die oben genannten Algorithmen auf ihre Funktionsfähigkeit getestet wurden. Die erhaltenen Ergebnisse werden in Abschnitt 9.3 präsentiert. Abschnitt 9.4 schließt mit einer Zusammenfassung ab.

9.1 Architektur der Anwendung

Das Ziel der Lokalisierungsanwendung ist es, die von den einzelnen Knoten ermittelten Positionen an zentraler Stelle zu sammeln. Da dies insbesondere auch in einem mobilen Netz funktionieren soll, wurde zum Verteilen der Positionen ein Flooding-Algorithmus eingesetzt. Somit erhalten potenziell alle Knoten im Netz die Positionen aller anderen Knoten. Da alle in der Wiselib enthaltenen Routingalgorithmen die gleiche Schnittstelle aufweisen, sind durch Änderung der Konfiguration beliebige Weiterleitungsstrategien verwendbar.

Die Anwendung wurde als generische Wiselib-Applikation realisiert. Das bedeutet, dass der gleiche Quelltext unverändert für verschiedene Zielsysteme kompiliert werden kann. Dazu implementiert die Applikation eine spezielle Initialisierungsmethode (`init(...)`) in der alle verwendeten Klassen instanziiert werden. Die jeweilige Plattform wird dabei im Makefile gesetzt (über das Makro `OSMODEL`). Darüber wird dann über eine Typdefinition die zu verwendende Betriebssystemklasse (z.B. `iSenseOsModel` oder `ContikiOsModel`) auf den Typen `Os` abgebildet. Somit kann der Quelltext der Applikation betriebssystemunabhängig implementiert werden. Das folgende Listing zeigt einen Ausschnitt aus der (`init(...)`) Methode der Lokalisierungsanwendung.

```
1 void init(Os::AppMainParameter& value)
2 {
3     ...
4     radio_ = &wiselib::FacetProvider    <Os,
          Os::ExtendedLQIRadio>::get_facet(value);
5     radio2_ = &wiselib::FacetProvider<Os,
          Os::Radio>::get_facet(value);
6     position_ = &wiselib::FacetProvider<Os,
          Os::Position>::get_facet(value);
```

```
7       ...
8   }
```

In diesem Fall werden zwei Instanzen der Funkschnittstelle gespeichert. Diese unterscheiden sich im Funktionsumfang. Während `radio_` vom Typ `Os::ExtendedLQIRadio` ist, weist `radio2_` die Schnittstelle `Os::Radio` auf. Der Unterschied ist, dass `radio_` zusätzlich zu den empfangenen Paketen noch die Signalstärke mitliefert. Die Daten über die empfangene Signalstärke ist einerseits wichtig für die Lokalisierungsalgorithmen und andererseits wurde sie dazu verwendet, um Pakete mit einem geringeren LQI als LQI_{MIN} zu verwerfen bevor sie an die Lokalisierungsalgorithmen weitergereicht werden. Zum Weiterleiten von Daten soll jedoch jeder Knoten unabhängig von konkreten LQI Werten herangezogen werden. Daher wurden zwei verschiedene Instanzen zur Funkkommunikation mit verschiedenen Eigenschaften genutzt. Es ist jedoch zu beachten, dass dies eine reine Abstraktion auf Anwendungsebene darstellt und beide Instanzen die gleiche physikalische Funkschnittstelle verwenden.

Dies dient dazu, dass nur Pakete mit hohem LQI zur Lokalisierung verwendet werden, da nur so eine gute Lokalisierungsleistung erreicht werden kann (vgl. Abschnitt 8.6). Die simulative Evaluation der ausgewählten Lokalisierungsverfahren hat gezeigt, dass die Genauigkeit der Lokalisierung steigt, je näher die Knoten beieinander positioniert sind. Um eine hohe Genauigkeit zu erreichen, müssen die Knoten allerdings so nah positioniert werden, dass je nach Szenario ein Großteil der Knoten direkt verbunden ist. Es würde sich in diesem Fall also nicht mehr um ein Multi-Hop Netz handeln. In diesem Projekt werden aber ja grundsätzlich und gerade auch im Rahmen der Sicherheitsarchitektur Multi-Hop Netze betrachtet. Es ergibt sich somit ein gewisser Ziel-Konflikt zwischen der Genauigkeit der Lokalisierung und der im Laboraufbau gewünschten und für ein Sensornetz typischen Multi-Hop Eigenschaft. Das vorgenannte Verfahren führt nun dazu, dass aus Sicht der nicht an der Lokalisierung beteiligten Verfahren ein Multi-Hop Netzwerk aufgespannt wird. Die Lokalisierungsalgorithmen verwenden jedoch nur sehr nah benachbarte Knoten, um die Lokalisierung durchzuführen.

Zusätzlich zu der Initialisierung der Funkschnittstelle ist auch die Parametrisierung der Lokalisierungsmodule erwähnenswert. Das folgende Listing zeigt, wie die Module für Sum-Dist und Min-Max parametrisiert werden. Es werden Referenzen zu den zu verwendenden Instanzen für die Funkschnittstelle, dem Uhrzeitmodul, der Debugschnittstelle, einer `shared_data_` Instanz und einem Distanzschätzungsmodul übergeben. Danach wird das Lokalisierungsmodul parametrisiert und die wesentlichen Parameter sind die zu verwendenden Instanzen für die Multi-Hop Distanzschätzung (hier: Sum-Dist), die Positionsberechnung (hier: Min-Max) und das Refinement (hier: ohne Funktion). Während der Refinement-Phase wird nach der Positionsberechnung versucht unter Einbeziehung der Positionen von benachbarten Knoten, die eigene Positionsschätzung zu verbessern. Da dies jedoch häufig zu einer Fehlerpropagation im gesamten Netzwerk führt, wurde diese Phase nicht weiter betrachtet.

```
sum_dist_module_.init(*radio_, *clock_, *debug_, shared_data_,
    distance_);
min_max_module_.init(*radio_, *debug_, shared_data_);
localization_.init(*radio_, *timer_, *debug_, shared_data_,
    sum_dist_module_, min_max_module_, nop_module_);
```

Die `shared_data_` Instanz enthält Informationen über den eigenen Knoten und benachbarte Knoten. Es werden darin Daten, wie zum Beispiel die aktuelle Position, ob der Knoten selbst ein Anker ist, benachbarte Knoten, bekannte Anker, Entfernung zu Ankern, etc. gehalten. Ein Teil der Daten wird a priori konfiguriert während andere Daten (z.B. die eigene Position falls es kein Ankerknoten ist) von den Modulen gesetzt werden. So ist insbesondere das Ergebnis der Lokalisierung enthalten. Das folgende Listing zeigt, wie die Daten abhängig von der eigenen Knoten-ID gesetzt werden. Der Knoten mit der ID $0x3$ ist ein Ankerknoten und erhält daher eine bekannte Position und einen hohen Konfidenzwert während Nicht-Ankerknoten zu Beginn eine unbekannte Positionsschätzung erhalten.

```
...
if (radio_->id()  ==  0x3)
{
    shared_data_.set_anchor(true);
    shared_data_.set_confidence(1.0);
    wiselib::Vec<Arithmetic> position(0, 20, 0);
    shared_data_.set_position( position );
    ...
}
else
{
    shared_data_.set_anchor(false);
    shared_data_.set_confidence(0.1);
    shared_data_.set_position(Os::Position::UNKNOWN_POSITION);
}

shared_data_.set_floodlimit(4);
shared_data_.set_check_residue(false);
shared_data_.set_idle_time(9000);
...
```

Zur Schätzung der Single-Hop Distanzen dient die Klasse `LQIDistanceModel`. Diese erwartet zur Konfiguration eine Tabelle, die LQI Werte auf Distanzen abbildet. Zur korrekten Parametrisierung werden der `init()` Methode die zu verwendende Instanz der Funkschnittstelle, eine Debuginstanz und eine `LQIMap` übergeben. Der letztgenannte Parameter enthält die vorgenannte Abbildung von LQI Werten auf Distanzen. Da diese Abbildung stark von der Umgebung und der Hardware abhängt, muss diese für den konkreten Anwendungsfall erstellt und der Instanz übergeben werden. Das folgende Listing zeigt beispielhaft, wie diese Abbildung erstellt werden kann.

```
...
lqi_dist_map.insert(wiselib::pair<lqi_t,Arithmetic>(151,2.0));
lqi_dist_map.insert(wiselib::pair<lqi_t,Arithmetic>(119,4.0));
...
```

Die verwendeten Daten für die Abbildung von LQI Werten auf Distanzen stammen aus den Messwerten, die in Abschnitt 8.1 beschrieben wurden. Die Werte sind in Abbildung 9.1 tabellarisch zusammengefasst. Wird ein LQI Wert empfangen, der nicht in der Tabelle enthalten ist, so wird interpoliert.

Distanz [m]	LQI Mittelwert [m]	LQI Standardabweichung [m]
2,00	151,456	4,965
4,00	119,405	1,794
6,00	104,715	2,988
8,00	89,139	2,531
10,00	77,317	4,021
12,00	73,393	4,600
14,00	59,608	6,703
16,00	55,535	6,649
18,00	46,127	9,317
20,00	42,660	8,135
22,00	51,245	10,777
24,00	42,826	7,681
26,00	44,058	10,196
28,00	42,866	9,083
30,00	33,888	9,517
32,00	31,006	9,744
34,00	25,449	9,140
36,00	24,244	7,235
38,00	12,000	9,200
40,00	23,900	9,200
42,00	24,000	9,200

Abbildung 9.1: Aus Messwerten gewonnene Abbildung von LQI auf den Mittelwert der Distanz. Dargestellt ist ergänzend die ermittelte Standardabweichung.

Nach dem Ablauf einer gewissen Zeitspanne wird angenommen, dass ein Knoten seine Position berechnet hat. Zu diesen Zeitpunkten wird per Callback eine Methode namens `evaluate()` aufgerufen. Die Aufgabe dieser Methode ist es, die aktuell ermittelte Position über ein Routingprotokoll zu verteilen. Dazu wird zunächst ein Nachrichtenpaket zusammengestellt und dann gesendet.

```
1  void evaluate(void*)
2  {
3      //Only send position information if this is no anchor
4      if (shared_data_.is_anchor())
5          return;
6
7      //Create the message buffer
8      ...
9
10     //Transmit via flooding
11     flooding_routing.send(radio2_->BROADCAST_ADDRESS,
           debug_message_length,  (Os::Radio::block_data_t*)
           debug_message_);
12 }
```

Danach wurden die Algorithmen zurückgesetzt und die Positionsberechnung startet von neuem. Nach erneutem Ablauf der Zeitspanne wird dann wieder die Methode `evaluate` aufgerufen und der Prozess startet von vorn.

Zu Beginn wird über einen sogenannten Gatewayknoten eine Nachricht in das Netzwerk geflutet. Daraufhin starten alle Knoten die Lokalisierungsanwendung. Der jeweilige Startzeitpunkt wird dabei zufällig um eine kurze Zeitspanne verzögert, damit nicht alle Knoten zur gleichen Zeit starten. Damit wird die Wahrscheinlichkeit von Kollisionen gesenkt. Der Gatewayknoten empfängt im Folgenden die oben genannten Pakete mit Informationen über die Knoten-ID, die berechnete Position und einen Index, der angibt, die wievielte Berechnungsrunde erreicht wurde. Jedes erhaltene Paket wird über die serielle Schnittstelle ausgegeben und von einem angeschlossenen PC empfangen und zur späteren Auswertung gespeichert.

9.2 Testumgebung

Die Lokalisierungsanwendung wurde im WISEBED Testbed getestet. Derzeit sind neun europaweit verteilte Testbeds im WISEBED-Verbund föderiert. Für diesen Bericht wurde jedoch nur das Testbed in Lübeck verwendet, das derzeit aus 162 Knoten besteht. Das Testbed ist in den Räumlichkeiten des Instituts für Telematik untergebracht und besteht aus 54 Clustern mit jeweils einem iSense, TelosB und Pacemate Sensorknoten (vgl. Abbildung 9.2). Die Verteilung der einzelnen Cluster ist in Abbildung 9.3 dargestellt, wobei jede Zahl die ID des jeweiligen iSense Knotens im Cluster darstellt. Die rot markierten und unterstrichenen Knoten sind die für die Evaluation verwendeten Ankerknoten.

Abbildung 9.2: Cluster bestehend aus einem iSense, TelosB und Pacemate Knoten.

Der Wert von LQI_{MIN} für das LQI-basierte Verwerfen von Nachrichten war auf 60 gesetzt. Dieser Wert beeinflusst maßgeblich die Anzahl der benachbarten Knoten, die zur Lokalisierungsberechnung herangezogen werden. Nachrichten von Knoten, die mit einem LQI-Wert $lqi < LQI_{MIN}$ empfangen werden, werden

Abbildung 9.3: Topologie des WISEBED Testbeds am Institut für Telematik der Universität zu Lübeck (dargestellt sind die IDs der iSense Knoten des jeweiligen Clusters).

verworfen. Betrachtet man Abbildung 9.1, so ergibt sich für $LQI_{MIN} = 60$, dass nur Nachrichten von maximal $15m$ entfernten Knoten berücksichtigt werden (sofern die gleichen Ausbreitungsbedingungen gelten wie beim Ermitteln der Werte aus der Abbildung).

9.3 Ergebnisse

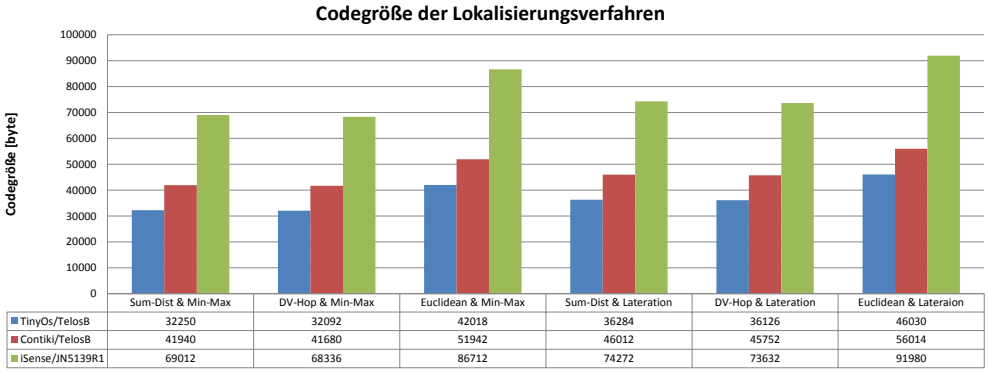
In diesem Abschnitt werden Evaluationsergebnisse präsentiert, die im Zusammenhang mit der Portierung der Lokalisierungsalgorithmen auf echte Hardwareplattformen und der Entwicklung der Lokalisierungsanwendung aufgenommen wurden. Die hier präsentierten Daten stellen eine erste Evaluation der Funktionsfähigkeit der Implementierung dar und sind insbesondere nicht als eine ausführliche Diskussion der Leistungsfähigkeit der Lokalisierung zu verstehen.

Auf dem WISEBED Testbed wurden vier mögliche Kombinationen von Lokalisierungsverfahren evaluiert. Hierbei wurden die beiden Positionsbestimmungsverfahren Min-Max sowie Lateration mit den Multi-Hop Distanzschätzern SumDist und Distance-Vector-Hop (DV-Hop) kombiniert, so dass die folgenden Verfahren untersucht wurden:

- SumDist mit MinMax
- DV-Hop mit MinMax

- SumDist mit Lateration
- DV-Hop mit Lateration

Abbildung 9.4 stellt die Codegröße der oben genannten Kombinationen dar. Dargestellt ist jeweils die resultierende Größe des Betriebssystems, der Anwendung und dem verwendeten Lokalisierungsverfahren. Im Vergleich zum Min-Max Verfahren benötigt der Lateration Algorithmus aufgrund der höheren Komplexität mehr Speicher. Auf der iSense Plattform beträgt dieser Unterschied in etwa 6kByte.



	Sum-Dist & Min-Max	DV-Hop & Min-Max	Euclidean & Min-Max	Sum-Dist & Lateration	DV-Hop & Lateration	Euclidean & Lateraion
TinyOs/TelosB	32250	32092	42018	36284	36126	46030
Contiki/TelosB	41940	41680	51942	46012	45752	56014
iSense/JN5139R1	69012	68336	86712	74272	73632	91980

Abbildung 9.4: Codegröße der verschiedenen Lokalisierungsverfahren.

Tabelle 9.1 und Abbildung 9.5 stellen die Ergebnisse von 100 Wiederholungen jeder Kombination der Lokalisierungsverfahren dar. Hierbei sind die Ergebnisse getrennt für die insgesamt 4 Lokalisierungsverfahren nebeneinander auf der X-Achse aufgetragen. Die Y-Achse zeigt den Positionsfehler in Metern. Dargestellt sind jeweils der minimale und maximale Positionsfehler in allen 100 Wiederholungen (oberer und unterer Whisker), der Median (Linie in der Mitte der Box) sowie das untere und obere Quartil (die Box) für jedes Verfahren. Bei der Interpretation der Ergebnisse ist zu beachten, dass das WISEBED Testbed in einem Gebäude installiert ist und somit eine unbekannte Dämpfung der Signale erfolgt. Die Applikation auf den Knoten verwendet jedoch eine Abbildung von LQI-Werten zu Distanzen, die auf Freifeldmessungen basiert. Somit sind nicht die absoluten Werte von Bedeutung sondern das Verhältnis zwischen den einzelnen Algorithmen.

Aus den aufgenommenen Daten ist ersichtlich, dass sich bei der Verwendung von Min-Max trotz unterschiedlicher Verfahren zur Multi-Hop Distanzschätzung (Sum-Dist und DV-Hop) die besseren Ergebnisse erzielen lassen. Abgesehen von deutlich stabileren Ergebnissen beträgt die Standardabweichung bei der Verwendung von Min-Max $2 - 5m$ während bei der Verwendung von Lateration die Standardabweichung $7 - 13m$ beträgt. Im WISEBED Testbed ließen sich mit der Kombination aus Sum-Dist und Min-Max die besten Ergebnisse erzielen. Der Median beträgt etwa $3.5m$, das Minimum liegt bei $0m$ und das

Maximum bei ca. $12m$. Zu erkennen ist ebenfalls, dass 50% der Sensorknoten einen Positionsfehler zwischen $2m$ und $6m$ aufweisen.

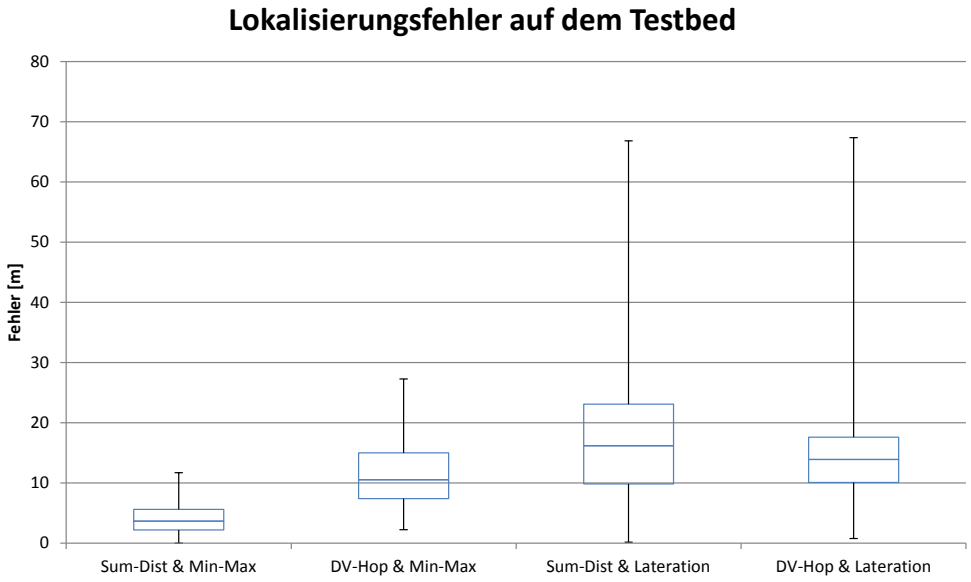


Abbildung 9.5: Lokalisierungsfehler im WISEBED Testbed (iSense Knoten).

	Sum-Dist Min-Max	DV-Hop Min-Max	Sum-Dist Lateration	DV-Hop Lateration
Max	11.71 m	27.28 m	66.85 m	67.36 m
3. Quartil	5.63 m	15 m	23.07 m	17.62 m
Median	3.66 m	10.51 m	16.19 m	13.92 m
1. Quartil	2.2 m	7.42 m	9.85 m	10.10 m
Min	$\leqslant$0.1 m	2.24 m	0.17 m	0.75 m

Tabelle 9.1: Wertetabelle der Ergebnisse der Lokalisierung in Szenario 1

9.4 Zusammenfassung

In diesem Abschnitt wurde die implementierte Lokalisierungsanwendung vorgestellt. Ein wesentlicher Bestandteil der Anwendung sind die Lokalisierungsalgorithmen, die plattformunabhängig für die Wiselib vorliegen. Nach derzeitigem Stand lassen sich die Anwendung und die Lokalisierungsalgorithmen im Simulator Shawn und auf den Hardwareplattformen iSense und TelosB und mit den Betriebssystemen iSense, TinyOS und Contiki übersetzten. Theoretisch möglich ist die Übersetzung für weitere, von der Wiselib unterstützten, Plattformen. Dies wurde jedoch im Rahmen dieser Arbeiten nicht weiter getestet.

10 Experimentelle Evaluation von Lokalisierungsverfahren

Dieses Kapitel dokumentiert die Implementierung und experimentelle Evaluation von Lokalisierungsverfahren. Abschnitt 10.1 stellt den Feldtest der Lokalisierungsanwendung und der Vitaldatenanwendung im Carlebach-Park vor. Anschließend werden in Abschnitt 10.2 die Ergebnisse der Messungen von Distanzen auf Basis der Signalumlaufzeit präsentiert.

10.1 Feldtest im Carlebach-Park

In Abschnitt 9 wurde die in WSNLAB implementierte Lokalisierungsanwendung vorgestellt. Das Ziel dieser Anwendung ist es, die von den einzelnen Knoten ermittelten Positionen an zentraler Stelle zu sammeln. Da dies insbesondere auch in einem mobilen Netz funktionieren soll, wurde zum Verteilen der Positionen ein Flooding-Algorithmus eingesetzt. Somit erhalten potenziell alle Knoten im Netz die Positionen aller anderen Knoten.

Die Anwendung wurde dahingehend erweitert, dass sie parallel zu den Positionsinformationen auch (falls verfügbar) die Herzfrequenz eines angeschlossenen Vitaldatensensors (vgl. Kapitel 11) überträgt. Somit sind an zentraler Stelle sowohl die geschätzten Positionen aller Knoten verfügbar als auch die Herzfrequenz mobiler Einsatzkräfte. Zur Erlangung der hier vorgestellten Ergebnisse wurde die iSense Plattform verwendet, da nur diese auch über die Hardware zur Messung der Herzfrequenz verfügt. Als Lokalisierungsverfahren wurde wie bereits in Abschnitt 9 Sum-Dist in Kombination mit Min-Max eingesetzt. Die Anwendungslogik und sonstige Parameter blieben unverändert (vgl. Abschnitt 9).

10.1.1 Versuchsaufbau und -durchführung

Der Feldtest fand im Freien im Carlebach-Park in Lübeck statt, der genaue Messort ist in Abbildung 10.1 durch ein Rechteck gekennzeichnet. Auf dieser Abbildung ist ebenfalls die Position der Videokamera markiert, die das Experiment von oben aufgezeichnet hat. Ein Foto des Experimentaufbaus im Carlebach-Park ist in Abbildung 10.2 abgebildet.

Die Verteilung der einzelnen Knoten ist schematisch in Abbildung 10.3 dargestellt. Das verwendete Areal hatte eine Ausdehnung von $24m * 48m$. Auf

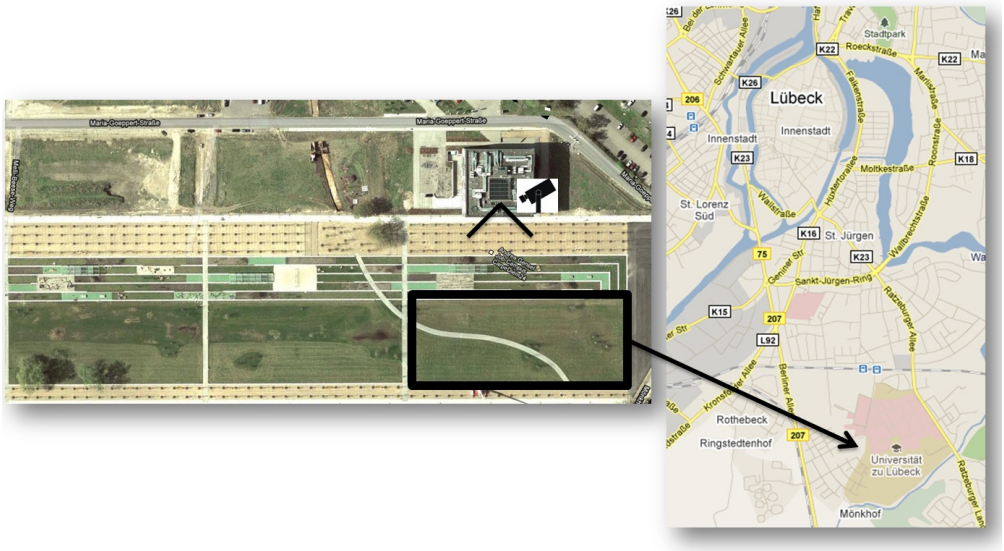


Abbildung 10.1: Messort im Carlebach-Park in Lübeck.

Abbildung 10.2: Foto des Experimentaufbaus im Carlebach-Park.

diesem Areal wurden 28 iSense Knoten in 20cm Höhe über Grund ausgebracht. Die einzelnen Knoten waren in einem Abstand von $8m$ aufgestellt, sodass die maximale Funkdistanz zwischen zwei Knoten $11,3m$ betrug. Die rot markierten Knoten fungierten als Ankerknoten mit bekannter Position. Abbildung 10.4

zeigt Identifikationsnummern der im Versuchsaufbau dargestellten Sensorknoten.

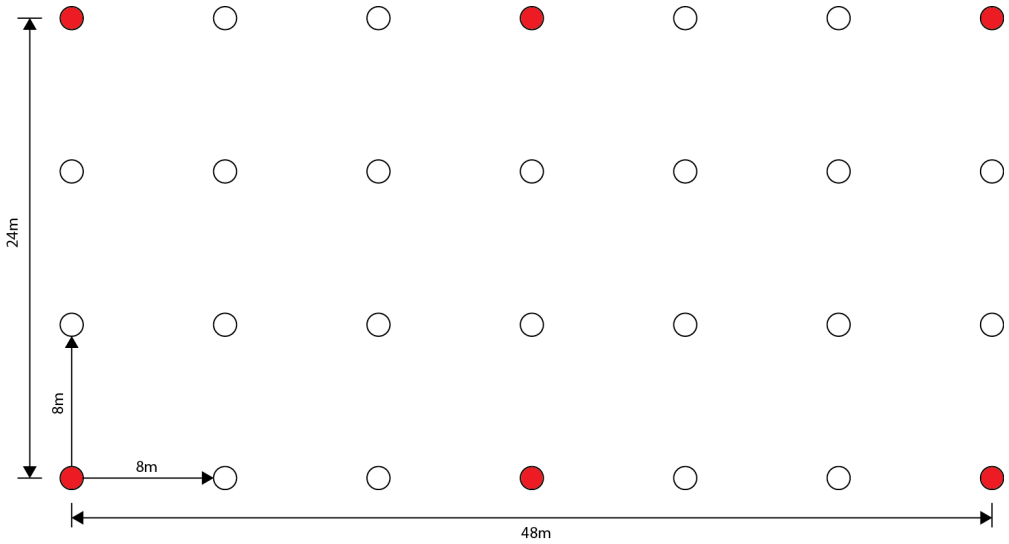


Abbildung 10.3: Schematische Darstellung des Versuchsaufbaus.

0x0404	0x0808	0x1212	**0x1616**	0x2020	0x2424	**0x2828**
0x0303	0x0707	0x1111	0x1515	0x1919	0x2323	0x2727
0x0202	0x0606	0x1010	0x1414	0x1818	0x2222	0x2626
0x0001	0x0505	0x909	**0x1313**	0x1717	0x2121	**0x2525**

Abbildung 10.4: Identifikationsnummern der im Versuchsaufbau dargestellten Sensorknoten. Ankerknoten sind rot hervorgehoben.

Die in Abbildung 10.3 abgebildeten Knoten befinden sich während der Laufzeit des Experimentes an einer festen Position. Nach einer Einschwingphase beginnt eine Einsatzkraft das Netzwerk zu durchschreiten. Die hier gewählte V-förmige Route ist in Abbildung 10.5 dargestellt. Während die mobile Einsatzkraft sich bewegt, errechnen alle im Netzwerk befindlichen Knoten (inkl. dem an der Einsatzkraft angebrachten Sensorknoten) periodisch die eigene Position und fluten diese im gesamten Netzwerk. Dies schließt den von der Einsatzkraft getragenen Knoten ein.

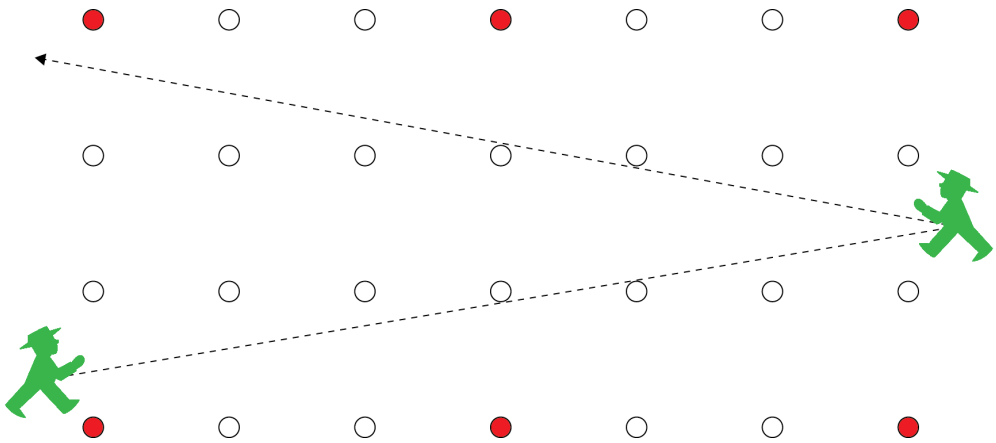


Abbildung 10.5: Versuchsdurchführung mit mobiler Einsatzkraft

Die gefluteten Daten werden von einem an ein Laptop angeschlossenen Sensorknoten empfangen und an diesen weitergeleitet. Dort werden die Daten gespeichert und von SpyGlass visualisiert. Abbildung 10.6 zeigt einen Auszug aus dem fertig editierten Video des Feldtests bei dem die Aufnahme der Videokamera und die Ausgabe von SpyGlass überlagert wurde. Zur besseren Erkennbarkeit wurde die Position der Knoten und der Anker durch Symbole hervorgehoben. Abbildung 10.7 zeigt eine vergrößerte Darstellung von SpyGlass mit zusätzlichen Erläuterungen. Zu sehen sind die realen Positionen der Sensorknoten und deren geschätzte Position sowie die geschätzte Position der Einsatzkraft inklusive der aktuellen Herzfrequenz.

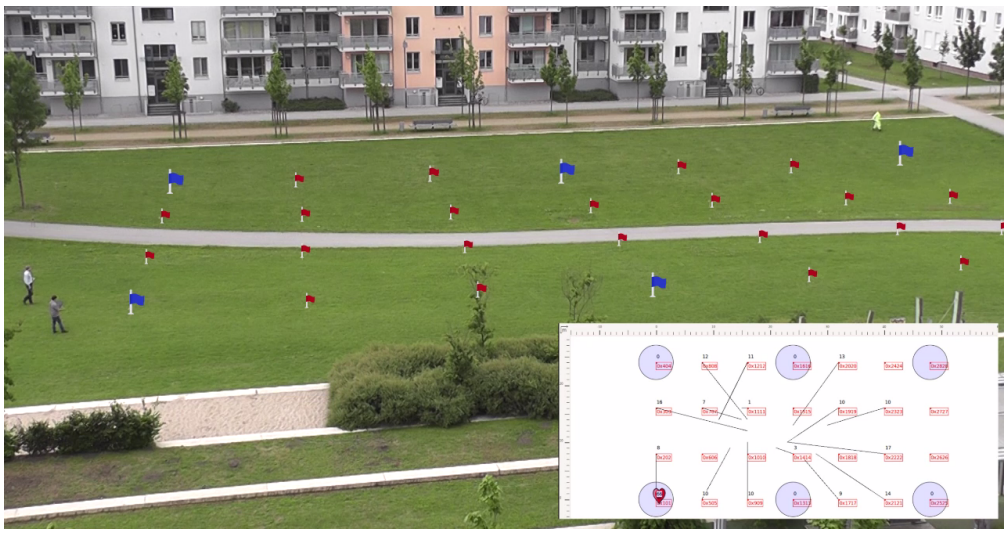


Abbildung 10.6: Auszug aus dem Video des Feldtests.

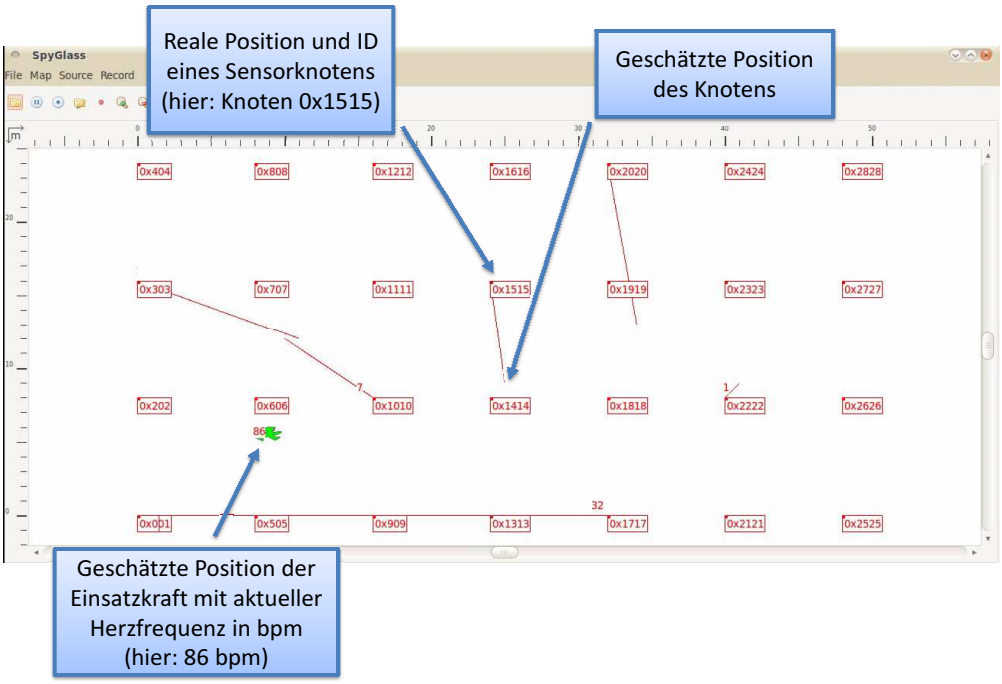


Abbildung 10.7: Auszug aus der Darstellung in SpyGlass.

10.1.2 Ergebnisse

Abbildung 10.8, Abbildung 10.9, Abbildung 10.10 und Abbildung 10.11 fassen die Ergebnisse der Reihen 1-4 in Form von Boxplots zusammen. Die Ankerknoten sind in den Grafiken enthalten, weisen aufgrund der als bekannt angenommenen Position jedoch keinen Fehler auf. Die Boxplots fassen die Ergebnisse von sieben Versuchsreihen zusammen.

Insgesamt lässt sich feststellen, dass die Ergebnisse bei den einzelnen Knoten sehr unterschiedlich ausfallen. Auf Basis der aufgenommen Daten lässt sich nur schwer ein einheitliches Fazit erstellen. Gemeinsam ist den Ergebnissen jedoch, dass eine Lokalisierung nur mit relativ großen Fehlern möglich scheint. Über die sieben aufgenommenen Messreihen gemittelt ergibt sich für den Lokalisierungsfehler ein Minimalwert von $6.39m$, ein Median von $11.69m$ und ein Maximalwert von $19.70m$.

Normalisiert man die Ergebnisse auf eine Kommunikationsreichweite von $40m$, so ergibt sich ein Lokalisierungsfehler zwischen 16% und 49%. Dies deckt sich im Wesentlichen mit den Ergebnissen aus Abschnitt 8.5 wo ein Lokalisierungsfehler bis 55% der Kommunikationsreichweite simulativ ermittelt wurde.

Damit lässt sich im Bilde der simulativen wie auch der praktischen Erprobung feststellen, dass sich die Multi-hop-Lokalisierung auf Basis von Signalstärkenmessungen nur sehr bedingt zur Bestimmung präziser Positionsinformationen eignet. Die tatsächlichen Gegebenheiten eines realen Szenarios lassen sich nur

mit hohem Aufwand a priori feststellen und modellieren. Darüber hinaus haben Umgebungsbedingungen, wie z.B. die Luftfeuchtigkeit, einen Einfluss auf die konkreten Messergebnisse und verfälschen damit die Ergebnisse der Distanz- und Positionsschätzung.

10.2 Distanzschätzung auf Basis der Signalumlaufzeit

Die bisherigen Untersuchungen im Projekt WSNLAB haben gezeigt, dass der wesentliche Einlußfaktor für die Genauigkeit der Lokalisierung die Single-Hop-Distanzmessung ist.

Da aus Sicht der Autoren ausschließlich Verfahren, die nur auf Basis der bereits existierenden Funkschnittstelle arbeiten, zum Einsatz in Sensornetzen geeignet sind, wurde diesen besondere Aufmerksamkeit geschenkt. Im bisherigen Projektverlauf wurde deutlich, dass die Distanzmessung mittels Signalabschwächung deutliche Defizite aufweist: Aufgrund des kubischen Verhaltens der Signalausbreitung ist nur bei geringen Distanzen ein starker Abfall der gemessenen Signalstärke mit zunehmender Distanz erkennbar. Bei größeren Distanzen ist die Abnahme der Signalstärke zunehmend geringer, so dass die Messungen immer störungsanfälliger werden. Dabei ist die Anzahl der Störquellen vielfältig, an vorderster Stelle sind hier Überlagerungen durch Signalreflektionen sowie Dämpfungen durch Hindernisse zu nennen.

Daher wurde im Projekt WSNLAB eine weitere Methode zur Single-Hop-Distanzschätzung untersucht: die Signalumlaufzeit. In diesem Bereich gab es kürzlich eine Neuentwicklung: der kombinierte Funk- und Controllerchip Jennic JN5148, der unter anderem auch auf dem Core Module 2 (CM20X) der coalesenses GmbH eingesetzt wird, bietet als momentan einziger zum Standard IEEE 802.15.4 kompatibler Funkchip eine Distanzschätzung mit Hilfe von Signalumlaufzeitmessungen. Obwohl im Rahmen von WSNLAB aus momentaner Sicht keine CM20X Module eingesetzt werden, sondern auf die älteren Grundmodule CM10X zurückgegriffen werden muss, wurden Distanzmessungen mit Hilfe des JN5148 durchgeführt, um die erzielbaren Genauigkeiten zu untersuchen.

Bei der Distanzschätzung auf Basis der Signalumlaufzeit versendet ein Gerät ein Signal und startet eine Zeitmessung. Das empfangende Gerät versendet sofort eine Antwort. Kommt diese beim ursprünglichen Absender an, stoppt dieser die Zeitmessung. Aus der Gesamtlaufzeit kann dann bei bekannter Ausbreitungsgeschwindigkeit der Signale der zurückgelegte Weg ermittelt werden. Die Schwierigkeit besteht hier darin, dass sich die Signale mit Lichtgeschwindigkeit bewegen. Somit ist eine extrem feingranulare Zeitmessung erforderlich, die technisch hoch anspruchsvoll ist. Insbesondere muss die Verarbeitung und Rücksendung beim empfangenden Gerät innerhalb einer genau bekannten Verarbeitungszeit geschehen, damit diese im Nachhinein herausgerechnet werden kann.

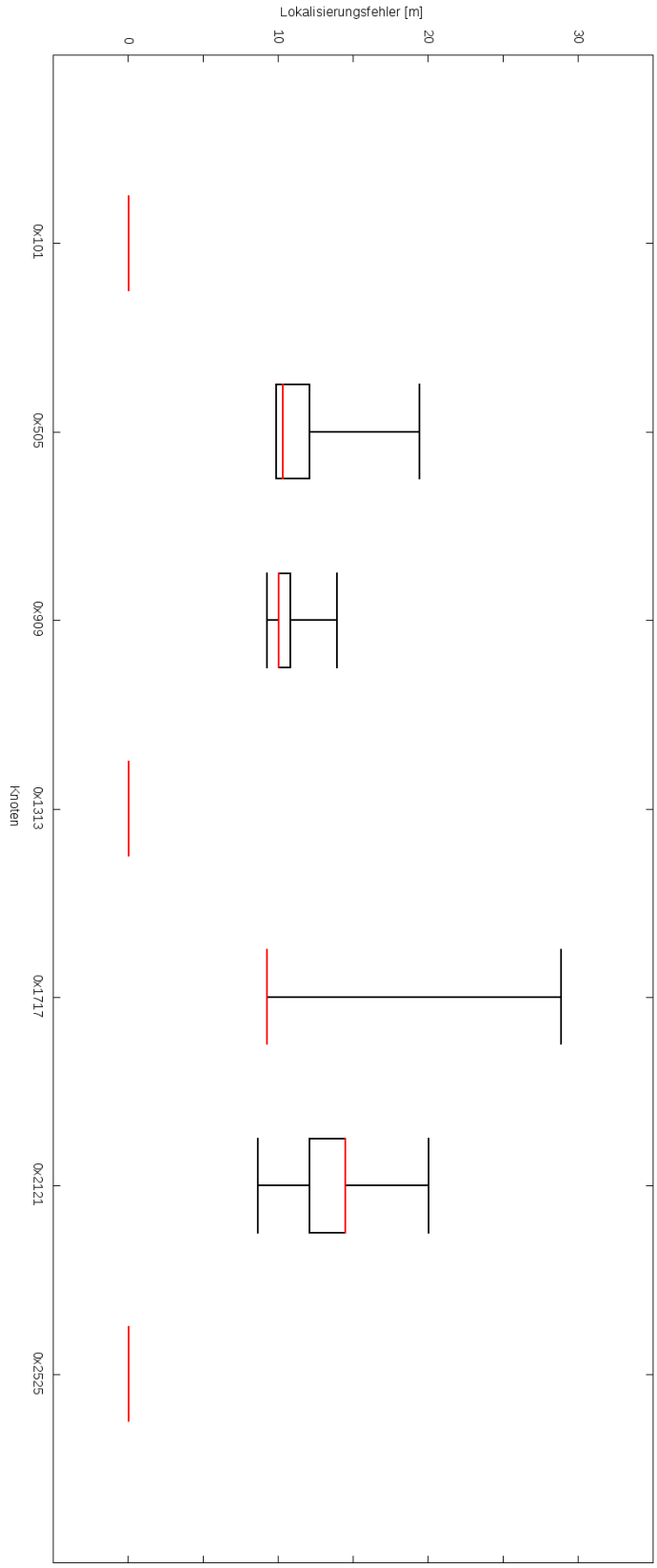


Abbildung 10.8: Boxplot der Ergebnisse der Reihe 1 (oberste Reihe).

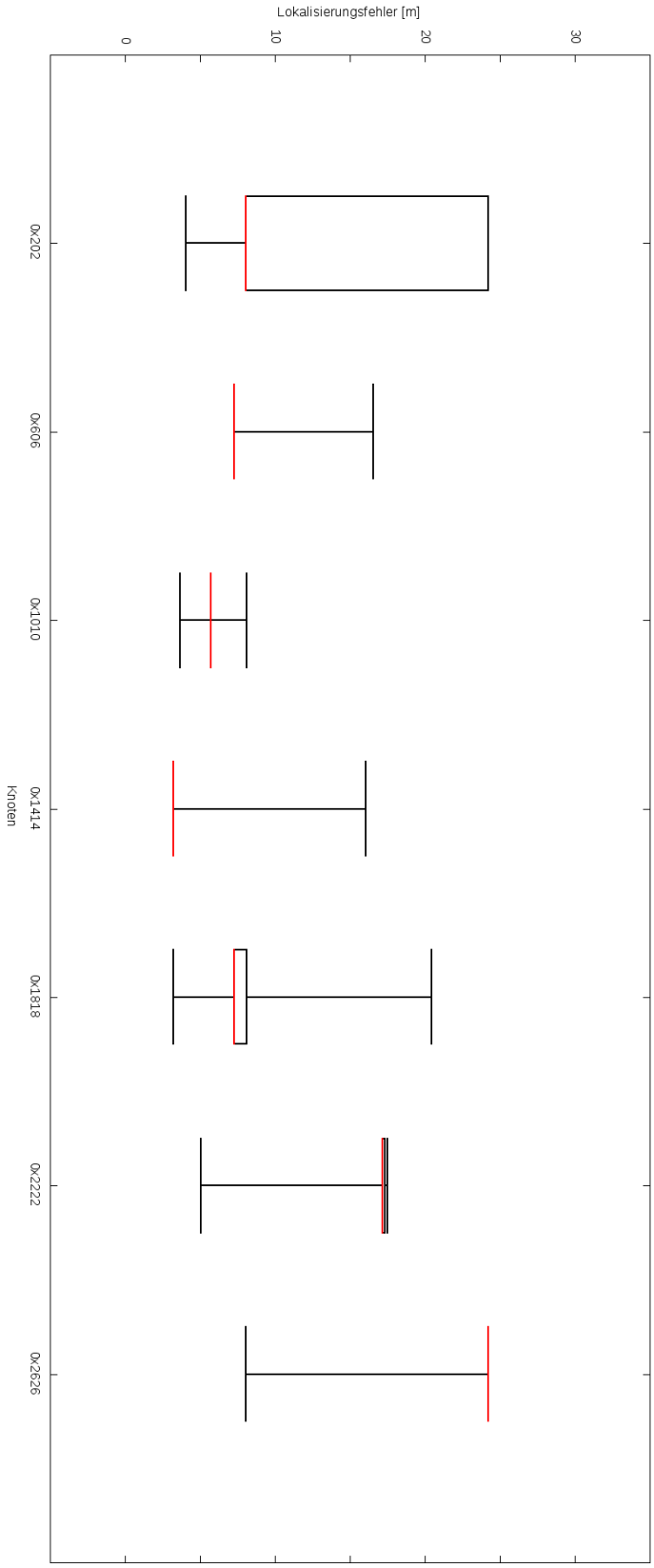


Abbildung 10.9: Boxplot der Ergebnisse der Reihe 2.

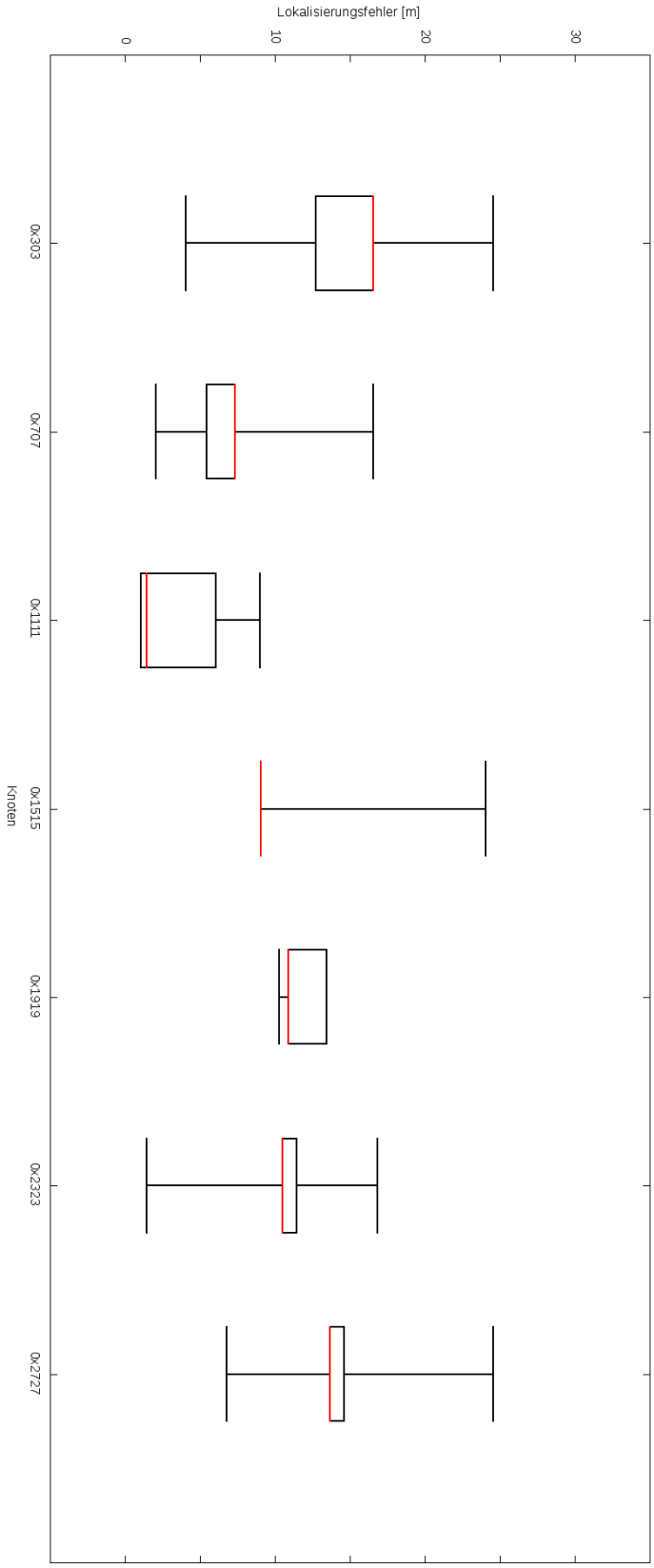


Abbildung 10.10: Boxplot der Ergebnisse der Reihe 3.

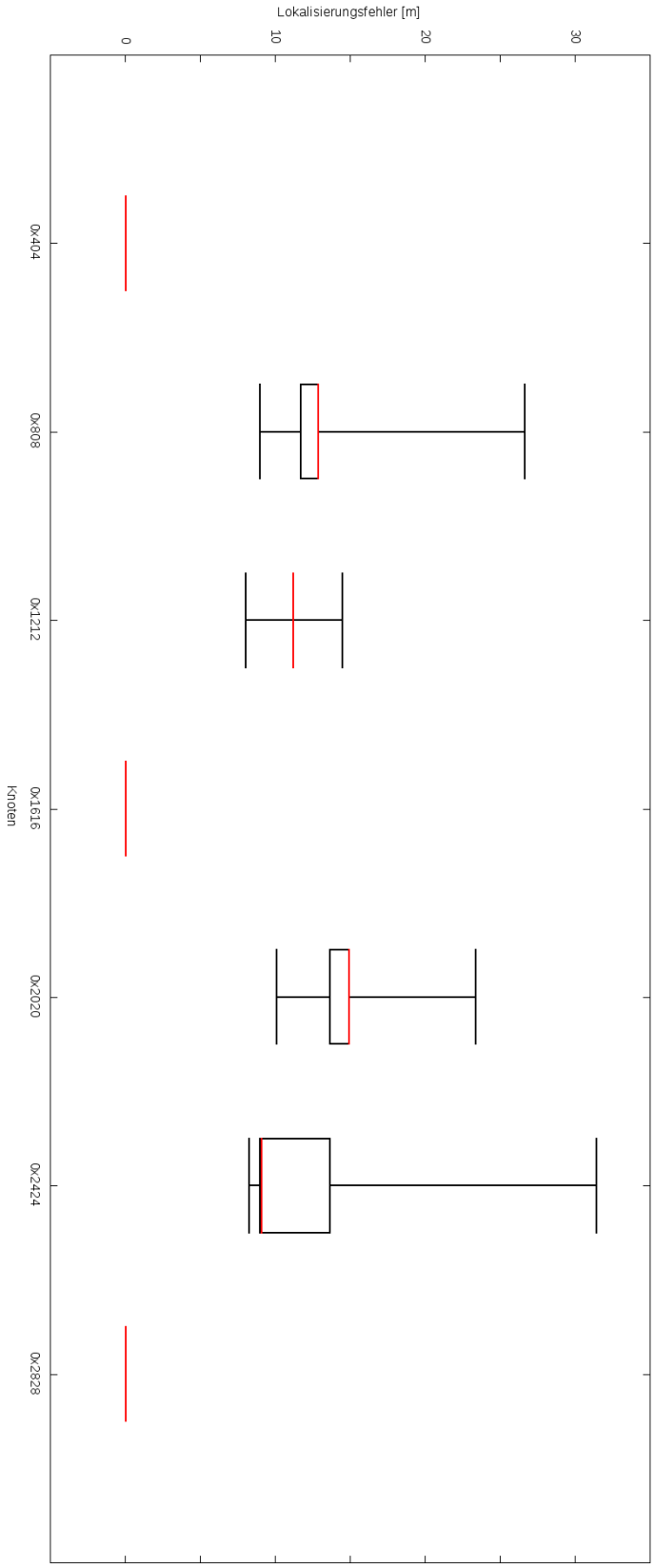


Abbildung 10.11: Boxplot der Ergebnisse der Reihe 4 (unterste Reihe).

Eine konzeptionelle Schwäche des Verfahrens ist, dass nicht die Distanz zwischen zwei Geräten gemessen wird, sondern die Länge des Signalweges. Ist der direkte Weg zwischen den Geräten durch ein Hindernis versperrt, wird das Signal einen entsprechend längeren Weg mittels Reflektion wählen, so dass dann dieser längere Weg zur Distanzmessung verwendet wird.

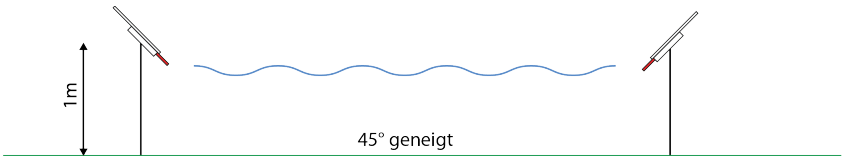
10.2.1 Versuchsaufbau

Die Versuche zur Bestimmung der Distanz zwischen zwei Sensorknoten mittels Signalumlaufzeitmessung fanden im Freien im Carlebach-Park in Lübeck statt, der genaue Messort ist in Abbildung 10.12 durch ein rotes Rechteck gekennzeichnet.

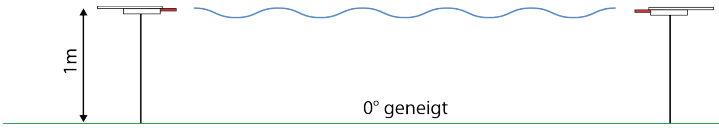
Abbildung 10.12: Messort im Carlebach-Park in Lübeck.

Als Hardware kamen zwei Sensorknoten bestehend aus einem coalesenses Grundmodul 2 mit integrierter Antenne (CM20I) und einem Batteriefach (PM10AA) zum Einsatz.

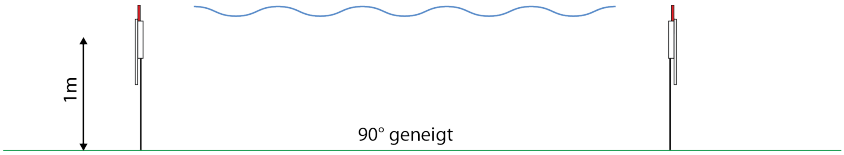
Für den Versuchsaufbau wurden die Sensorknoten jeweils auf einem Stativ in einer Höhe von einem Meter befestigt, die Messungen wurden mit den Abständen 10m, 20m, 40m und 70m durchgeführt. Um die Auswirkung der Geräteanordnung auf die Distanzschätzung zu untersuchen, wurde für jede Distanz die Messung in drei unterschiedlichen Neigungen der Knoten (0°, 45° und 90°) zueinander durchgeführt (Abbildung 10.13).



(a) Ausrichtung 45°



(b) Ausrichtung 0°



(c) Ausrichtung 90°

Abbildung 10.13: Anordnung der Sensorknoten für die Distanzmessungen

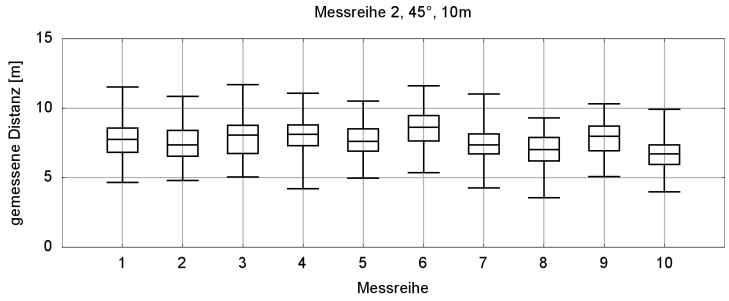
Während ein Großteil der Tests unter idealen Bedingungen stattfand, d.h. mit direkter Sichtverbindung der Geräte zueinander, wurden ebenfalls Messungen mit einer Person durchgeführt, die sich während der Messungen direkt vor einem der beiden Knoten befand.

Für jede Kombination aus Distanz und Neigung wurden 10 Messreihen mit jeweils 100 Einzelmessungen durchgeführt. Die im folgenden Abschnitt gezeigten Ergebnisse, dargestellt als sogenannte Boxplots, zeigen für diese 10 Messreihen Minimum, Maximum und Median der gemessenen Einzelwerte sowie die Quartilspreizung als Box. Diese Box beschreibt somit denjenigen Bereich, in dem die mittleren 50% der Messwerte liegen.

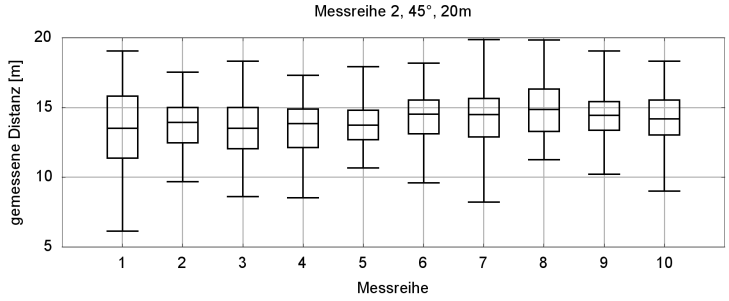
10.2.2 Ergebnisse

Die Abbildung 10.14 zeigt die Messergebnisse für die Abstände 10m, 20m, 40m und 70m bei einer Neigung von 45°, wie in Abbildung 10.13(a) dargestellt. Auf der x-Achse sind die 10 Messreihen aufgetragen, die y-Achse zeigt die jeweils gemessenen Distanzen.

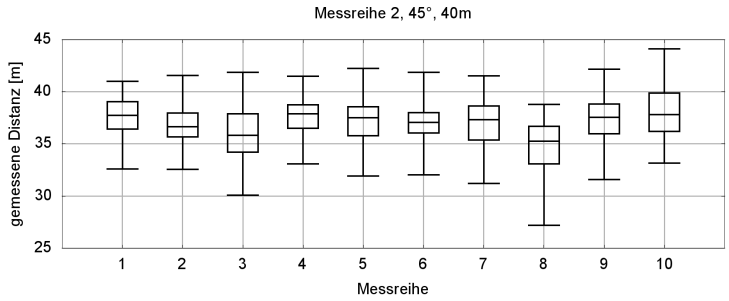
Zu erkennen ist, dass sich der Median der Messwerte bei allen der jeweils 10 Messreihen über die 4 Distanzen unterhalb der tatsächlichen Entfernung befindet. Besonders deutlich ist dies bei einer Distanz von 70 Meter (vergleiche Abbildung 10.14(d)) zu erkennen. Hier liegt der Median im Mittel 16% unter der tatsächlichen Distanz, was in diesem Fall 11,24 Metern entspricht. Für 40 Meter beträgt diese Abweichung 7,4% (2,69 Meter), für 20 Meter 29,55% (5,91 Meter), bei 10 Metern sind es 23,6% (2,36 Meter). Es lässt sich also weder ein



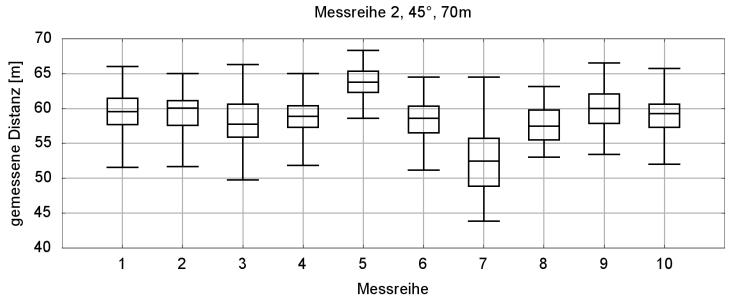
(a) 10m



(b) 20m



(c) 40m



(d) 70m

Abbildung 10.14: Median, Quartile und Extrema der Distanzschätzungen bei 45° Neigung und direkter Sichtverbindung

direkter Zusammenhang zwischen der tatsächlichen Distanz und dem Maß der Unterschätzung ableiten, noch ist die Unterschätzung konstant.

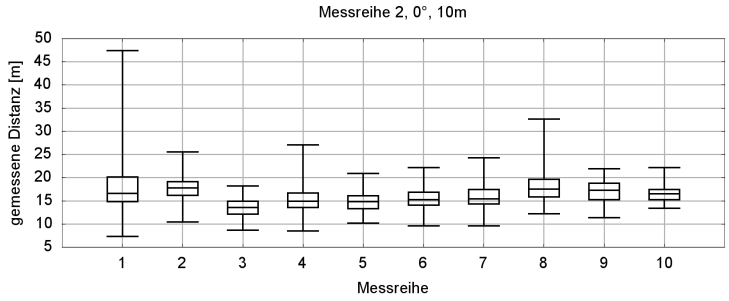
Weiterhin ist zu erkennen, dass mit steigender Distanz die Spreizung sowohl zwischen Minimum und Maximum als auch zwischen unterer und oberer Quartilgrenze zunimmt. Sie beträgt im Mittel bei 10 Metern 6,19 Meter (Min/Max) bzw. 1,68 Meter (Quartile), bei 20 Metern 9,35 Meter bzw. 2,75 Meter, bei 40 Metern 10,11 Meter bzw. 2,90 Meter und bei 70 Metern 13,63 Meter bzw. 4,06 Meter. Diese zunehmende Streuung spiegelt sich auch in mit zunehmender Distanz stärkeren Abweichungen zwischen den 10 Messreihen wieder.

Um den Einfluss der Antennenausrichtung auf die Distanzmessungen zu untersuchen, wurden ebenfalls Messungen mit Antennenausrichtungen von 0° und 90° durchgeführt. Abbildung 10.15 zeigt die ermittelten Ergebnisse bei einer Neigung der Knoten von 0°, wie in Abbildung 10.13(b) dargestellt. Wieder sind auf der x-Achse die 10 Messreihen aufgetragen, die y-Achse zeigt die jeweils gemessenen Distanzen.

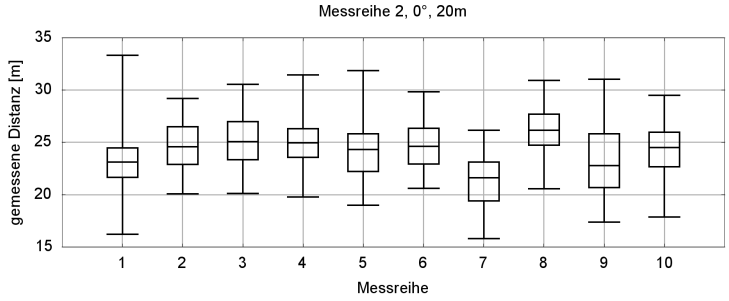
Es wird deutlich, dass bei dieser Messung keine klare Tendenz zur Überschätzung vorliegt. Während bei einer Distanz von 10 Metern der Median mit Mittel 59,2% (5,92 Meter) und bei 20 Metern 20,78% (4,15 Meter) über der tatsächlichen Distanz liegt, kommt es bei 40 Metern zu einer recht deutlichen Unterschätzung von 27,9% (11,16 Meter). Bei 70 Metern hingegen ergibt sich wiederum eine leichte Überschätzung (12,57% bzw. 8,8 Meter).

Dieses uneinheitliche Bild setzt sich bei der Streuung der Werte fort. Während sie prinzipiell auch hier mit der Distanz zunimmt, finden sich immer wieder recht deutliche Abweichungen bei einzelnen Messreihen. Dies gilt insbesondere für eine Distanz von 10 Metern (Messreihen 1,2,4,7,8), aber auch für Messreihe 1 bei 20 Metern oder die Reihen 1, 3 und 10 bei 70 Metern. Die mittlere Spreizung zwischen Minimum und Maximum sowie zwischen unterer und oberer Quartilgrenze beträgt bei 10 Metern 16,11 Meter (Min/Max) bzw. 3,25 Meter (Quartile), bei 20 Metern 11,64 Meter bzw. 3,50 Meter, bei 40 Metern 20,27 Meter bzw. 4,55 Meter und bei 70 Metern 25,11 Meter bzw. 6,77 Meter.

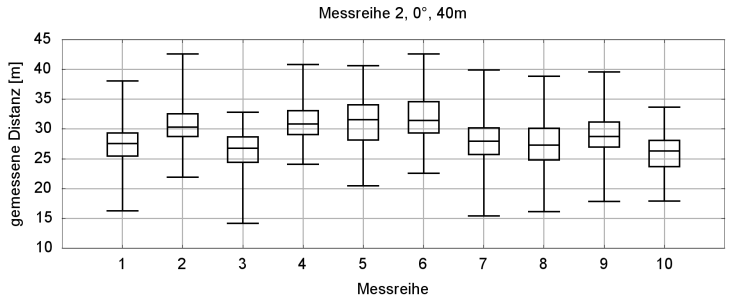
Das einheitlichste Bild zeigt sich bei den Messungen mit einer Antennenausrichtung von 90° (vergleiche Abbildung 10.13(c)). Die entsprechenden Ergebnisse sind in Abbildung 10.16 in der bekannten Form dargestellt. Zu erkennen ist, dass sich der Median der Messwerte bei allen der jeweils 10 Messreihen über die 4 Distanzen unterhalb der tatsächlichen Distanz befindet. Besonders deutlich ist dies bei einer Distanz von 40 Metern, bzw. 20 Metern (vergleiche Abbildung 10.16(c), bzw. Abbildung 10.16(b)) zu erkennen. Hier liegt der Median im Mittel 18%, bzw. 19% unter der tatsächlichen Distanz, was in diesem Fall 7,20 Metern, bzw. 3,67 Metern entspricht. Für 70 Meter beträgt diese Abweichung 4,74% (3,32 Meter), bei 10 Metern sind es 6,9% (0,69 Meter). Es lässt sich also weder ein direkter Zusammenhang zwischen der tatsächlichen Distanz und dem Maß der Unterschätzung ableiten, noch ist die Unterschätzung konstant.



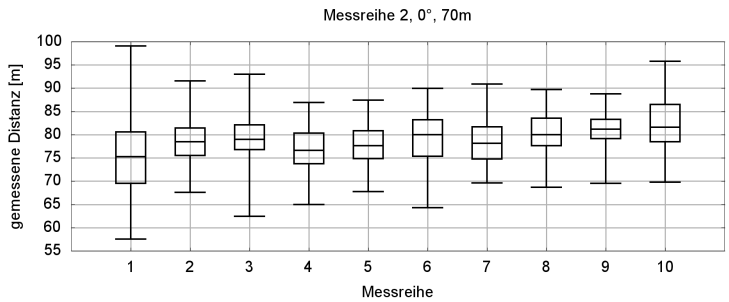
(a) 10m



(b) 20m

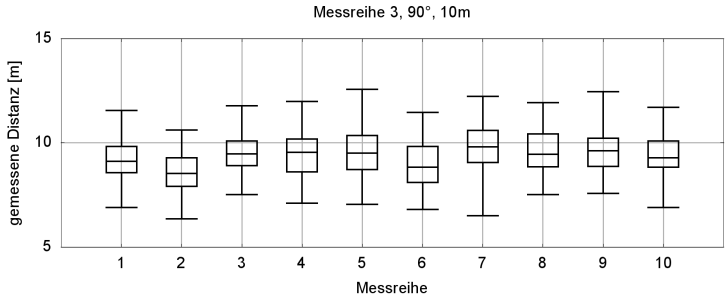


(c) 40m

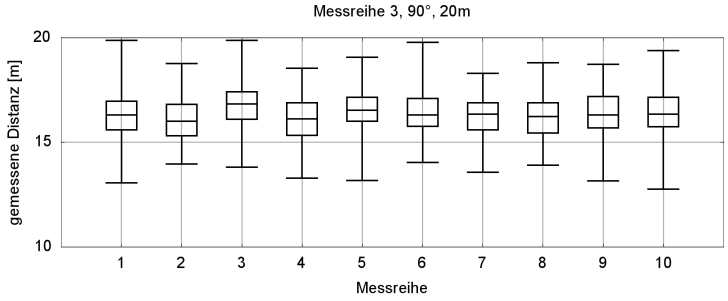


(d) 70m

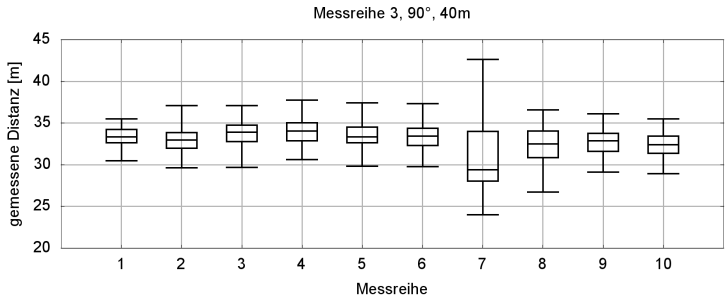
Abbildung 10.15: Median, Quartile und Extrema der Distanzschätzungen bei 0° Neigung und direkter Sichtverbindung



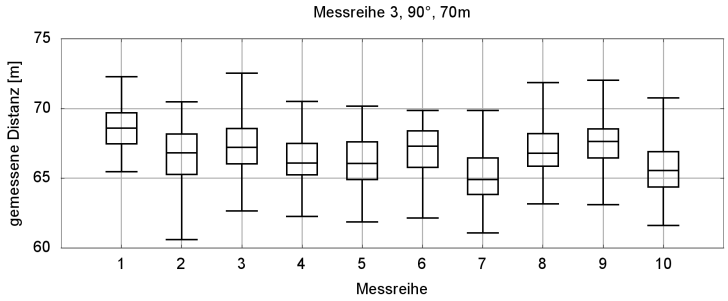
(a) 10m



(b) 20m



(c) 40m



(d) 70m

Abbildung 10.16: Median, Quartile und Extrema der Distanzschätzungen bei 90° Neigung und direkter Sichtverbindung

Weiterhin ist zu erkennen, dass mit steigender Distanz die Spreizung sowohl zwischen Minimum und Maximum als auch zwischen unterer und oberer Quartilgrenze zunimmt. Sie beträgt im Mittel bei 10 Metern 4,80 Meter (Min/Max) bzw. 1,44 Meter (Quartile), bei 20 Metern 5,63 Meter bzw. 1,38 Meter, bei 40 Metern 8,41 Meter bzw. 2,48 Meter und bei 70 Metern 8,63 Meter bzw. 2,48 Meter. Diese Streuung spiegelt sich auch in mit zunehmender Distanz leicht stärker werdenden Abweichungen zwischen den 10 Messreihen wieder.

Bei der siebten Messung über die Distanz von 40 Metern, abgebildet in Abbildung 10.16(c), sind im Vergleich zu den anderen Messungen deutlich höhere Abstände zwischen Minimum und Maximum (18,61 Meter) bzw. oberen und unteren Quartil (1,35 Meter) sichtbar. Die Ursache hierfür war eine Windböe, die während der Messung eines der Stative erfasste, aus seiner stationären Lage brachte und den Versuchsaufbau um etwa 30° kippte. An diesem Zwischenfall sieht man deutlich die Empfindlichkeit der durchgeführten Signalumlaufzeitmessungen gegenüber Veränderung der Ausrichtung.

Einfluss von Hindernissen im Signalweg

Da im Feld davon auszugehen ist, dass zwischen den Knoten keine permanente Sichtverbindung existiert, wurden zusätzlich Tests mit einem Hindernis in Form einer Person durchgeführt, die sich für den Verlauf der Messung permanent direkt vor dem die Messung initiierenden Knoten befand. Der Versuchsaufbau blieb gegenüber der vorangehend beschriebenen Anordnung unverändert, folgt der Darstellung in Abbildung 10.13(c), d.h. die Geräte waren im 90° Winkel zueinander ausgerichtet. Der Abstand zwischen den Geräten betrug 10 Meter.

Dabei befand sich die störende Person direkt vor einem der beiden Sensorknoten. Damit stellt diese Versuchsanordnung recht genau die Situation nach, wie sie sich in der ebenfalls im Rahmen von WSNLAB implementierten Vitaldatenanwendung darstellt: Die Personen tragen die Sensorgeräte direkt am Körper, so dass deren Kommunikationsbereich in einer Richtung nahezu komplett abgeschirmt ist.

Abbildung 10.17 zeigt die Messergebnisse mit der Person als Hindernis, die Vergleichswerte bei ungestörter Sichtverbindung können Abbildung 10.16(a) entnommen werden. Wieder sind auf der x-Achse die 10 Messreihen aufgetragen, die y-Achse zeigt die jeweils gemessenen Distanzen.

Es wird deutlich, dass die Einzelmessungen deutlich höheren Schwankungen unterliegen. Dieses schlägt sich insbesondere in den Minimal- und Maximalwerten nieder: der minimal gemessene Abstand betrug 0,01 Meter, das Maximum betrug 116,04 Meter. Ebenfalls angestiegen ist die Quartilspreizung, wenn auch weniger stark: der mittlere Abstand zwischen oberen und unteren Quartil betrug 5,47 Meter, während es ohne Hindernis 1,44 Meter waren.

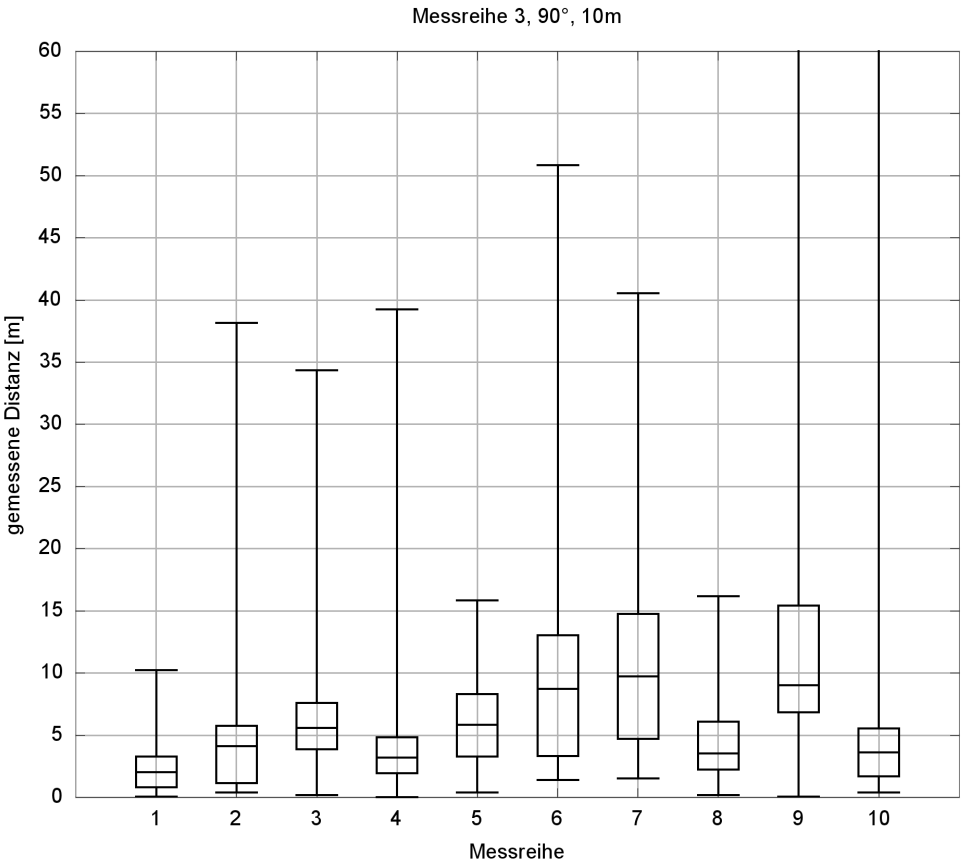


Abbildung 10.17: Median, Quartile und Extrema der Distanzschätzungen bei 10m Distanz und 90° Neigung mit einer Person im Signalweg. Die nicht abgebildeten Maxima der Messungen 9 und 10 betragen 116,04 Meter, bzw. 89,25 Meter.

Der Median der Messreihen lag im Mittel bei 5,52 Metern (was einer Unterschätzung von 44,8% entspricht), während es ohne Hindernis 9,31 Meter und somit eine Unterschätzung von 6,9% waren.

10.2.3 Zusammenfassende Bewertung

Die Auswertung der Messungen zeigt, dass sich das untersuchte Verfahren zur Distanzabschätzung auf Basis der Signalumlaufzeit nur bedingt für eine genaue Bestimmung der Entfernung zwischen zwei Knoten eignet.

Einerseits wurde deutlich, dass die Ausrichtung der Antennen zueinander einen nicht unerheblichen Einfluss zu haben scheint. Die beste Genauigkeit wurde bei paralleler Antennenausrichtung erzielt (90°, vergleiche Abbildung 10.13(c)), die schlechteste bei kollinearer Ausrichtung (0°, Abbildung 10.13(c)).

Bei direkter Sichtverbindung zwischen den Knoten und paralleler Antennenausrichtung konnte bei Wahl des Medians aus 100 Messungen eine Genauigkeit von

etwa 27% der Distanz (Messreihe 7 in Abbildung 10.16(c)) oder besser realisiert werden. Damit liegt die Genauigkeit zwar besser als bei Signalstärkenmessungen, es konnte jedoch keine Verbesserung von mehreren Größenordnungen erreicht werden.

Unterbricht zusätzlich eine Person die direkte Sichtbarkeit der Knoten, verursacht dies unter Umständen extreme Verschlechterungen der Genauigkeit. Legt man wiederum den Median aus 100 Messungen zugrunde, sind in diesem Fall Fehler von bis zu 80% der tatsächlichen Distanz aufgetreten (Messreihe 1 in Abbildung 10.17). Dieses ist sicher zu einem guten Anteil darauf zurückzuführen, dass die Störung hier besonders massiv war. Es wurde praktisch eine komplette Hälfte des Kommunikationsbereiches abgeschirmt. Insofern kann dieses Ergebnis als der schlechteste anzunehmende Fall angesehen werden. Er zeigt jedoch deutlich die Probleme auf, die von Hindernissen im Signalweg ausgehen.

Insgesamt lässt sich sagen, dass die Genauigkeit gegenüber Signalstärkenmessungen leicht verbessert werden konnte, insbesondere was die Reichweite angeht, über die Messungen möglich sind. Die verwendete Hardware der Firma Jennic ist jedoch noch recht neu am Markt, so dass ein gewisser Anlass zu der Hoffnung besteht, dass eine zunehmende Reife auch eine weitere Verbesserung der Messgenauigkeit mit sich bringen wird.

10.3 Fazit

Die in diesem Kapitel vorgestellten Untersuchungen haben gezeigt, dass sich in der Realität die Lokalisierung in Sensornetzen weiterhin schwierig gestaltet. Die bei Einsatz von Distanzschätzungen auf Basis von Signalabschwächung erzielbaren Genauigkeiten sind recht enttäuschend. Es können überhaupt nur bei kurzen Distanzen Schätzungen erzielt werden, bei denen Ergebnis und Abstand miteinander korrelieren. Daher wurden bei den Lokalisierungsexperimenten Pakete mit Signalstärken unterhalb einer gewissen Schranke verworfen, so dass überhaupt nur Kommunikation über kurze Distanzen stattfand. Doch selbst bei geringen Abständen zwischen benachbarten Sensorknoten sind die berechneten Positionen sehr ungenau.

Distanzschätzungen auf Basis der Signalumlaufzeit liefern bessere Ergebnisse als solche auf Basis der Signalabschwächung, insbesondere sind hier Schätzungen über die gesamte Kommunikationsreichweite möglich. Allerdings haben die Experimente auch gezeigt, dass die Messungen recht anfällig gegen Variationen des Umfeldes sind, insbesondere Hindernisse im Signalweg können leicht zu massiven Fehlern führen. Auch hat die Ausrichtung der Antennen zueinander starken Einfluss auf die Genauigkeit. Es bleibt abzuwarten, ob eine weitere Reifung dieser Technik in der Zukunft zu Verbesserungen in der Genauigkeit beitragen kann.

11 Vitaldatenerweiterung

Im Rahmen des vom Auftraggeber beauftragten optionalen Arbeitspaketes *Vitaldatensensor* wurde eine neue Hardwarekomponente für die iSense Sensornetzwerkplattform entwickelt.

Sie kann mit Hilfe eines Brustgurtes, wie er von Pulsuhren bekannt ist, die Herzfrequenz von Personen erfassen und an eine Sensornetzkomponente übermitteln, von wo aus die Daten mit Hilfe des Sensornetzes beispielsweise an eine Einsatzleitstelle weitergegeben werden können.

Es wurden 20 Prototypen entwickelt, gefertigt und im Projekt eingesetzt.

11.1 Hardware

Abbildung 11.1 zeigt den Schaltplan des neuen Moduls HRM10.

Es basiert auf dem Empfänger für Herzfrequenzbrustgurte Polar RMCM01. Dieses ist das einzige kommerziell erhältliche Modul, das die Signale der Polar-Brustgurte empfangen kann. Es stellt insofern eine gute Wahl dar, als das sich die Burstgurte von Polar als Quasistandard durchgesetzt haben, und nahezu alle am Markt erhältlichen Gurte zu ihnen kompatibel sind.

Der Empfänger benötigt an äußerer Beschaltung hauptsächlich einen Quarz, und stellt die Signale des Brustgurtes in Form von Pulsen an einem IO-Pin zur Verfügung. Der entsprechende Ausgang wurde an Pin 29 (S1_INT/DIO8) des 34-poligen Bussteckers X1 geführt. Nach einer Startup-Verzögerung von 5 Sekunden für kodierte Signale und 15 Sekunden für unkodierte Signale befindet er sich standardmäßig auf dem Low-Pegel, und wechselt für jeden Herzschlag kurz auf den High-Pegel. Der Polar-Empfänger verfügt darüber hinaus über einen weiteren Signalausgang, der ohne Verzögerung alle empfangenen Signale durch einen Puls anzeigt und an Pin 13 (S2_INT/DIO18) angeschlossen ist. Der Reseteingang des Empfängers ist mit Pin 27 (VGA_ON/DIO10) verbunden. Er arbeitet mit inverser Logik, erzeugt also bei Low-Pegel einen Reset.

Das HRM10 (siehe Abbildung 11.2) wurde so entwickelt, dass es in ein Gehäuse vom Typ 1593Pxx der Firma Hammond Manufacturing geschraubt werden kann. Die Anordnung wurde so gewählt, dass die bevorzugte Empfangsrichtung des Polar-Empfängers nach oben zeigt. Das Modul verfügt darüber hinaus über 2 mit den übrigen iSense Modulen kompatible M2 Bohrungen, durch die Schrauben zur Befestigung weiterer Module geführt werden können. In das Batteriefach des 1593Pxx wurde ein Batteriehalter für zwei AA Batterien eingelegt,

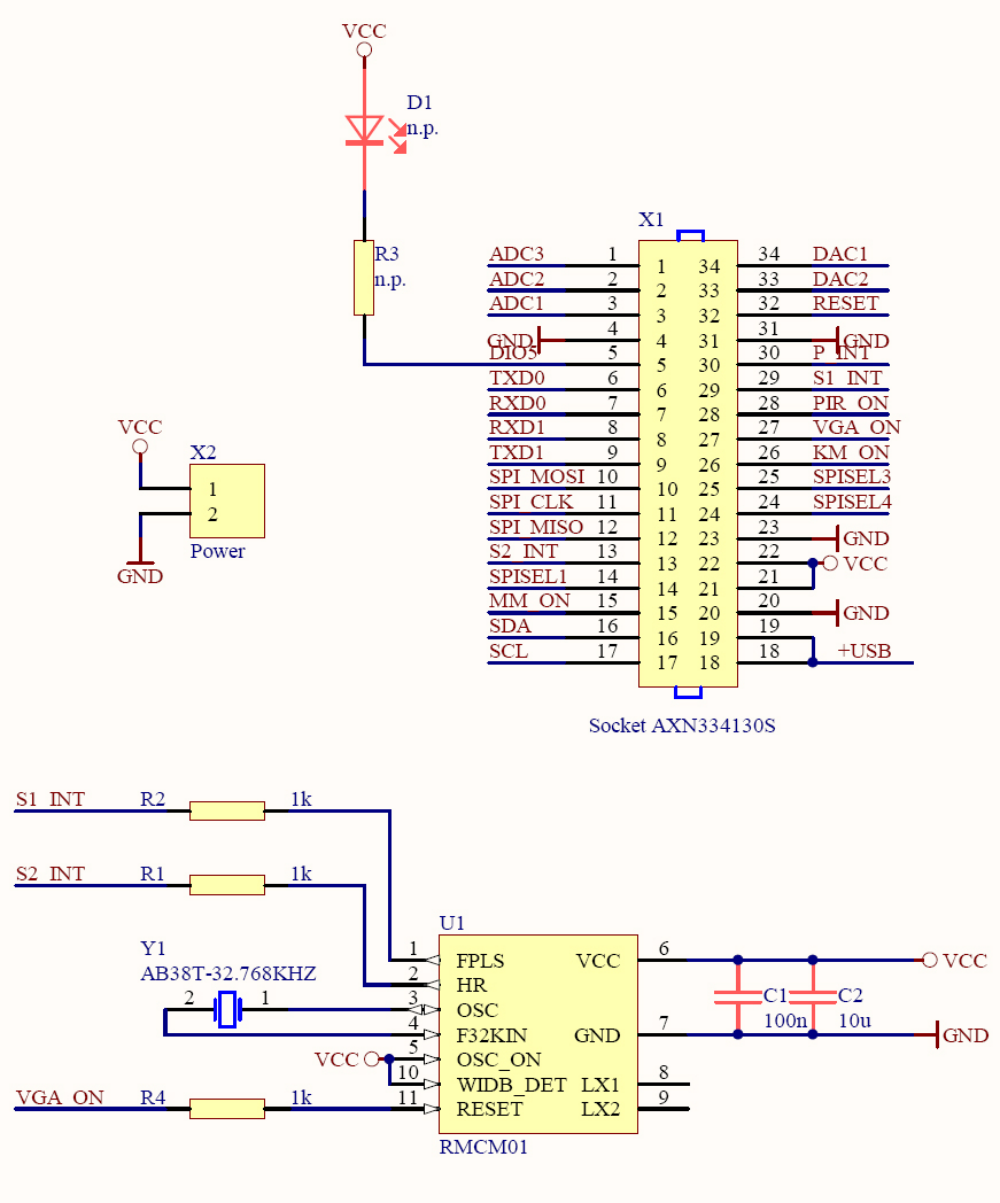


Abbildung 11.1: Schaltplan des Herzfrequenzempfangsmoduls HRM10.

der an die Einlötlöcher X2 des HRM10 angeschlossen wurde. Diese speisen die VCC Pins des Bussteckers (Pins 21/22), und können so weitere angeschlossene Module mit Strom versorgen.

Die im Projekt eingesetzten Sensorknoten bestehen aus der die zuvor beschriebenen Basis aus Gehäuse und HRM10 sowie iSense Grundmodulen der ersten Generation (CM10I), die vom BSI zur Verfügung gestellt wurden (Abbildung 11.3).

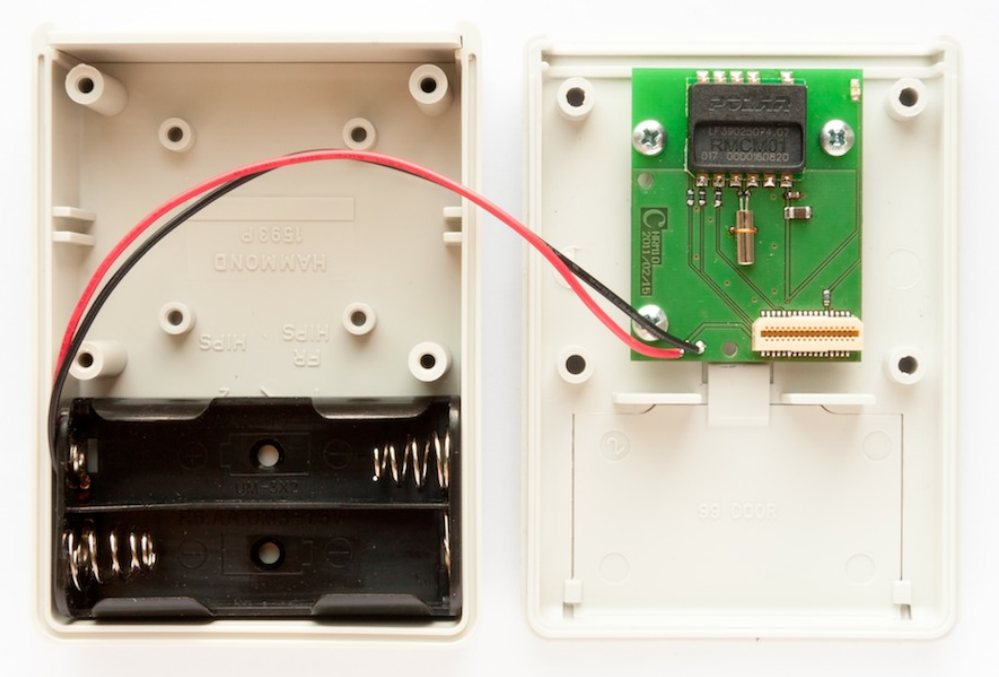


Abbildung 11.2: Herzfrequenzempfangsmodul HRM10 mit Gehäuse und Batteriefach.

11.2 Software

Für das neue Hardwaremodul HRM10 wurde eine Treiberklasse (Abbildung 11.4) entwickelt und in den Open-Source-Teil der iSense-Firmware integriert. Sie wertet die Pulse des Polar-Empfängers aus und berechnet die aktuelle Herzfrequenz.

Der Treiber bestimmt bei jedem Interrupt den zeitlichen Abstand zum vorigen Interrupt und berechnet so die aktuelle Herzfrequenz. Um die Robustheit gegenüber Störungen des Polar-Empfängers zu erhöhen, kommt wie im Empfängerdatenblatt empfohlen eine Mittelwertbildung zum Einsatz.

Dabei wird der jeweils aktuelle Interruptabstand nur anteilig zur Bestimmung der Herzfrequenz herangezogen. Es kommt ein sogenannter Exponential-Weighted-Moving-Average-Filter (EWMA-Filter) zum Einsatz, da dieser nur eine geringe Berechnungs- und Speicherkomplexität aufweist. Er ermittelt das zur Berechnung der geglätteten Herzfrequenz verwendete Intervall zwischen zwei aufeinander folgenden Interrupts i_n gemäß der Formel

$$i_n = \alpha t_{int} + (1 - \alpha) * i_{n-1}$$

wobei t_{int} die Zeit zwischen dem aktuellen und dem vorangegangenen Interrupt bezeichnet. Für α wurde der Wert 0.1 gewählt. Da die Intervalle in Millisekunden angegeben werden, ergibt sich die Herzfrequenz gemäß der Formel

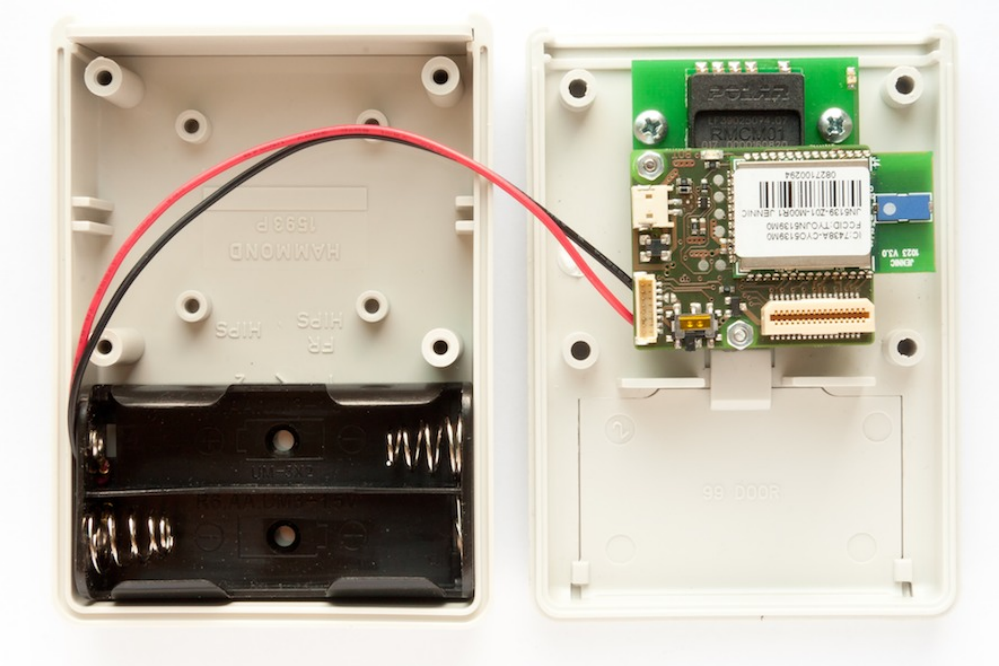


Abbildung 11.3: Vitaldatensensorknoten.

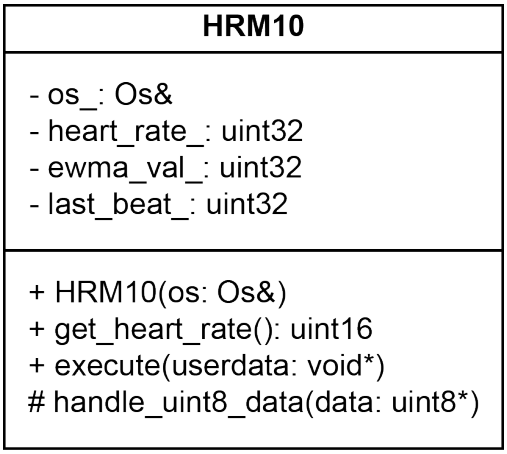


Abbildung 11.4: Treiberklasse für das Vitaldatenmodul.

$$f = \frac{60000}{i_n}$$

wobei f die Herzfrequenz widerspiegelt.

Wird innerhalb einer Zeit von zwei Sekunden kein Interrupt ausgelöst, so wird ein Kommunikationsabbruch zwischen Brustgurt und Empfänger angenommen. Daraufhin wird die Herzfrequenz auf 0 gesetzt. Nachdem zwei Interrupts innerhalb zweier Sekunden aufgetreten sind, beginnt die Berechnung der Herzfrequenz erneut.

Die Benutzung des Treibers für das HRM10 wird in einer kleinen Demonstrationsanwendung [37] verdeutlicht. Diese übertragt die Herzfrequenz per Funk an einen PC, der die Verlauf der Herzfrequenz(en) auf einem PC mit Hilfe der von iShell im *Curve Illustrator (Single Window)* [39] darstellt.

11.3 Zusammenfassung

Im Rahmen des Arbeitspaketes *Vitaldatensensor* wurde ein neues Hardwaremodul für die iSense Sensornetzwerkplattform entwickelt, das mit Hilfe eines Brustgurtes die Herzfrequenz von Personen erfassen kann.

Es wurde eine Treiberklasse implementiert und in den Open-Source-Teil der iSense Firmware integriert, die die Funktionen des Hardwaremoduls kapselt und die Herzfrequenzinformationen zugänglich macht.

Eine Demo-Anwendung dient der Demonstration der Funktionalität und erleichtert als Vorlage die Benutzung der Treiberklasse.

A Akronyme

ACK	Acknowledgement
ADV	Advertisement
AES	Advanced Encryption Standard
AM	Active Message
AoA	Angle of Arrival
API	Application Programming Interface
APIT	Approximate Point-In-Triangulation Test
BLIP	Berkeley Low-power IP stack
CAP	Contention Access Period
CBC-MAC	Cipher Block Chaining Message Authentication Code
CCA	Clear Channel Assessment
CCM	Counter mit CBC-MAC
CDMA	Code Division Multiple Access
CFP	Contention Free Period
CSMA	Carrier Sense Multiple Access
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance
CST	Carrier-Sense-Time
CTP	Collection Tree Protocol
CTR	Counter Modus
DAG	Directed Acyclic Graph
DIO	DODAG Information Object
DLP	Diskreter Logarithmus Problem
DODAG	Destination Oriented Directed Acyclic Graph
DoS	Denial-of-Service
DSA	Digital Signature Algorithm
EAD	Energy-Aware Data-centric routing
EAR	Energy-Aware Routing
ECC	Elliptic Curve Cryptography
ECDSA	Elliptic Curve DSA
EEPROM	Electrically Erasable Programmable Read-Only Memory
ETX	Expected Transmission Count
EUI	Extended Unique Identifier
EWMA	Exponentially Weighted Moving Average
FDMA	Frequency Division Multiple Access
FEC	Forward Error Correction
FFD	Full-Function Device
FPU	Floating-Point Unit
GPS	Global Positioning System
GUI	Graphical User Interface
IDS	Intrusion Detection System
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IP	Internet Protocol
IPv6	Internet Protocol Version 6
JRE	Java Runtime Environment
kbps	kilobits per second
LAN	Local Area Network
LBR	LLN Border Router
LCS	Link Control Server
LE	Link Estimation
LEACH	Low-Energy Adaptive Clustering Hierarchy
LEEP	Link Estimation Exchange Protocol
LLN	Low power and Lossy Network
LPL	Low Power Listening
LPP	Low Power Probing
LPS	Local Positioning System
LQI	Link Quality Indicator
LR-WPAN	Low-Rate Wireless Personal Area Network

MAC	Media Access Control
MANET	Mobile Ad-hoc NETwork
Mbps	Megabit per second
MCU	MicroController Unit
MDS	Multidimensionale Skalierung
MIPS	Million Instructions Per Second
MOAP	Multihop Over-the-Air-Programming
MoSe	Modellierung drahtloser Sensornetze
MTU	Maximum Transfer Unit
NACK	Negative ACK
NIST	National Institute of Standards and Technology
OS	Operating System
OSI	Open Systems Interconnection
OTAP	Over-The-Air Programming
P2P	Peer-to-Peer
PAN	Personal Area Network
PDR	Packet Delivery Ratio
PIR	Passive InfraRed
RAM	Random Access Memory
RDC	Radio Duty Cycling
REQ	Request
RF	Radio Frequency
RFD	Reduced-Function Device
ROLL	Routing Over Low power and Lossy networks
ROM	Read-Only Memory
RPL	IPv6 Routing Protocol for LLNs
RSSI	Received Signal Strength Indicator
RTS/CTS	Request To Send - Clear To Send
SDK	Software Development Kit
SDR	Software Defined Radio
SEC	Standards for Efficient Cryptography
SecOTAP	Secure OTAP
SHA	Secure Hash Algorithm
SPIN	Sensor Protocols for Information via Negotiation
TDMA	Time Division Multiple Access
TEP	TinyOS Enhancement Proposal
TSN	Trusted Sensor Node
UART	Universal Asynchronous Receiver Transmitter
UKoLoS	Ultra-Wideband Radio Technologies for Communications, Localization and Sensor Applications
UKW	Ultrakurzwelle
USRP	Universal Software Radio Peripheral
UWB	Ultra Wide Band
VOR	VHF Omnidirectional Radio Range
WISEBED	Wireless Sensor Network Testbeds
WLAN	Wireless Local Area Network
WSN	Wireless Sensor Network
WSNLAB	Wireless Sensor Network Lab
Xcast	Explicit Multicast

B Kurzfassung in englischer Sprache

A Wireless Sensor Network (WSN) consists of small, resource-constrained computing devices (so-called motes) that perform physical measurements (e.g., temperature, vibration) in a distributed manner. The motes form a self-adaptive multi-hop network to transport the measured data to a sink. The data may be pre-processed and fused in the network. Furthermore, WSNs often provide capabilities using a reverse channel for sensor control and management as well as flashing of motes and Over-The-Air Programming (OTAP). WSNs are deployed in a steadily growing plethora of application areas. Especially their deployment in the industrial, military, public safety, and medical domains renders security in these networks an issue of high relevance.

The main goal of the WSNLab project, partly funded by the German Federal Office for Information Security (BSI), is to build a WSN laboratory for the evaluation of WSN security measures. A second goal is to develop, implement, and evaluate a security architecture for stationary and mobile WSNs. By doing so, different hardware and software platforms are considered to reflect realism through heterogeneity. Thus, all evaluations are performed within a testbed consisting of multiple Operating Systems (OSs) and sensor platforms to ensure broad system support. As localization and (concerning security) dislocation are important aspects in WSNs, different localization algorithms are implemented and evaluated as well.

The heterogeneous testbed consists of three operating systems (Contiki [47], iSense [33], and TinyOS [74]) running on three hardware platforms (TelosB [109], MicaZ [108], and iSense-CM10C [38]). A simple collector application on the nodes collects and transmits the sensor data to the sink. As routing protocols, we run the Collection Tree Protocol (CTP) [66] and the IPv6 Routing Protocol for LLNs (RPL) [157]. Reconfigurations as well as the deployment of new protocols is possible using OTAP. To realize multi-hop paths in a reliable and reproducible way, the testbed supports topology management using a link control system based on the concept of virtual links (cf. [15]). Even complex topologies can be defined by manipulating the delivery of messages. The topologies can even be changed during an experiment as well. By doing so, the motes become virtually mobile and mobile sensor networks can be evaluated in the testbed.

An attacker can achieve his objective(s) through different kinds of attacks. These are categorized based on the targeted layer. The threat analysis is first divided into the goals of the attacker. A goal is further connected to the attacks that may be used to reach that specific goal. The attacks are grouped by the layer

on which they are conducted. The specific properties of WSNs lead to special attacks as well as new challenges for countermeasure development: The resource scarcity of the nodes in terms of computing power, memory, energy, and bandwidth requires countermeasures to be light-weight but also effective at the same time. We have implemented a security architecture based on such countermeasures. Preventive measures such as traffic encryption and digital signatures were implemented on the motes. The detection measures were integrated in an Intrusion Detection System (IDS). The design of the IDS is modular. Thus, further modules can be easily integrated. However, the resource scarcity of the nodes limits the number of modules that can be used simultaneously.

In addition to the security architecture, different ranging technologies and localization algorithms are simulated, implemented and experimentally evaluated. For single-hop ranging, distance estimation based on RSSI and time-of-flight are used and compared with regard to performance. Sum-Dist, DV-Hop and Euclidean schemes are employed to convert distances from single-hop to multi-hop. Min-Max and Lateration algorithms come into operation for position estimation. For the localization an application was implemented that also integrates the new heartbeat sensor module designed in the project.

Tabellenverzeichnis

Abbildungsverzeichnis

Literaturverzeichnis

[1] Adebutu, T., Sacks, L., and Marshall, I. Simple Position Estimation for Wireless Sensor Networks. In *Proc. of London Communications Symposium (LCS '03)* (London, England, 2003).

[2] Akkaya, K., and Younis, M. A Survey on Routing Protocols for Wireless Sensor Networks. *Ad Hoc Networks 3*, 3 (2005), 325–349.

[3] Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., and Cayirci, E. Wireless Sensor Networks: A Survey. *Computer Networks 38* (2002), 393–422.

[4] Al-Karaki, J. N., and Kamal, A. E. Routing Techniques in Wireless Sensor Networks: A Survey. *IEEE Wireless Communications 11*, 6 (2004), 6–28.

[5] Albowicz, J., Chen, A., and Zhang, L. Recursive Position Estimation in Sensor Networks. In *Proc. of 9th International Conference on Network Protocols (ICNP '01)* (Riverside, California, USA, 2001), pp. 35–41.

[6] Aschenbruck, N., Bauer, J., Bieling, J., Schwamborn, M., Martini, P., Pfisterer, D., Hakim, K., Fischer, S., and Buschmann, C. Projektbericht Wireless Sensor Networks-Labor (WSNLAB) - Meilenstein 2. Tech. rep., Universität Bonn, Universität Lübeck, coalesenses GmbH, Bonn, Germany, 2011.

[7] Aschenbruck, N., Bauer, J., Schwamborn, M., Martini, P., Pfisterer, D., Fischer, S., and Buschmann, C. Projektbericht Wireless Sensor Networks-Labor (WSNLAB) - Meilenstein 1. Tech. rep., Universität Bonn, Universität Lübeck, coalesenses GmbH, Bonn, Germany, 2010.

[8] Aschenbruck, N., Ernst, R., Gerhards-Padilla, E., and Schwamborn, M. BonnMotion - a mobility scenario generation and analysis tool. In *Proc. of the 3rd Int. ICST Conference on Simulation Tools and Techniques, Torremolinos, Spain* (2010), pp. 51:1–51:10.

[9] Aschenbruck, N., Gerhards-Padilla, E., and Martini, P. Simulative Evaluation of Adaptive Jamming Detection in Wireless Multi-hop Networks. In *Proc. of the 7th Workshop on Wireless Ad Hoc and Sensor Networks (WWASN '10)* (Genoa, Italy, 2010), pp. 213–220.

[10] Ash, J. N., and Potter, L. C. Robust System Multiangulation using Subspace Methods. In *Proc. of the 6th International Conference on Information Processing in Sensor Networks (IPSN '07)* (Cambridge, Massa-

chusetts, USA, 2007), pp. 61–68.

[11] Atmel Corporation. Atmel ATmega128L Data Sheet. http://www.atmel.com/dyn/resources/prod_documents/doc2467.pdf (Stand: 12.07.2011).

[12] Atmel Corporation. MCU Wireless – RZUSBstick. http://www.atmel.com/dyn/products/tools_card.asp?tool_id=4396 (Stand: 12.07.2011).

[13] Barker, E., Barker, W., Burr, W., Polk, W., and Smid, M. Sp 800-57: Recommendation for key management — part 1: General. *Nist Special Publication 800* (2007).

[14] Baronti, P., Pillai, P., Chook, V. W., Chessa, S., Gotta, A., and Hu, Y. F. Wireless Sensor Networks: A Survey on the State of the Art and the 802.15.4 and ZigBee Standards. *Computer Communications 30*, 7 (2007), 1655–1695.

[15] Baumgartner, T., Chatzigiannakis, I., Danckwardt, M., Koninis, C., Kröller, A., Mylonas, G., Pfisterer, D., and Porter, B. Virtualising Testbeds to Support Large-Scale Reconfigurable Experimental Facilities. In *Proc. of the 7th European Conference on Wireless Sensor Networks (EWSN '10)* (Coimbra, Portugal, 2010), pp. 210–223.

[16] Baumgartner, T., Chatzigiannakis, I., Fekete, S., Koninis, C., Kröller, A., and Pyrgelis, A. Wiselib: A Generic Algorithm Library for Heterogeneous Sensor Networks. In *Proc. of the 7th European Conference on Wireless Sensor Networks (EWSN '10)* (Coimbra, Portugal, 2010), pp. 162–177.

[17] Behnke, R., and Timmermann, D. AWCL: Adaptive Weighted Centroid Localization as an efficient improvement of Coarse Grained Localization. In *Proc. of 5th Workshop on Positioning, Navigation and Communication (WPNC '08)* (Hannover, Germany, 2008), pp. 243–250.

[18] Beigl, M., Budde, M., Sigg, S., Zitterbart, M., Hergenröder, A., Haas, C., and Dudek, D. Modellierung drahtloser Sensornetze (MoSe) - Abschlussbericht. Tech. rep., Universität Braunschweig, Universität Karlsruhe, Germany, 2010.

[19] Bergamo, P., and Mazzini, G. Localization in Sensor Networks with Fading and Mobility. In *Proc. of the 13th International Symposium on Personal, Indoor and Mobile Radio Communications* (Lisbon, Portugal, 2002), pp. 750–754.

[20] Berkeley WEBS – Wireless Embedded Systems. BLIP Tutorial. http://docs.tinyos.net/tinywiki/index.php/BLIP_Tutorial (Stand: 12.07.2011).

[21] Blumenthal, J., Grossmann, R., Golatowski, F., and Timmermann, D. Weighted Centroid Localization in Zigbee-based Sensor Net-

works. In *Proc. of the International Symposium on Intelligent Signal Processing (WISP '07)* (Alcala de Henares, Spain, 2007), pp. 1–6.

[22] Boivie, R., Feldman, N., Imai, Y., Livens, W., and Ooms, D. Explicit Multicast (Xcast) Concepts and Options (RFC 5058). `http://tools.ietf.org/html/rfc5058` (November 2007).

[23] Borg, I., and Staufenbiel, T. *Lehrbuch Theorien und Methoden der Skalierung*, 4 ed. Hübner, 2007.

[24] Boukerche, A., Cheng, X., and Linus, J. Energy-Aware Data-Centric Routing in Microsensor Networks. In *Proc. of the 6th International Workshop on Modeling Analysis and Simulation of Wireless and Mobile Systems (MSWiM '03)* (San Diego, California, USA, 2003), pp. 42–49.

[25] Brown, S., and Sreenan, C. Updating Software in Wireless Sensor Networks: A Survey. Tech. rep., University College Cork (UCC), Cork, Ireland, 2006.

[26] Buettner, M., Yee, G. V., Anderson, E., and Han, R. X-MAC: A Short Preamble MAC Protocol for Duty-Cycled Wireless Sensor Networks. In *Proc. of the 4th International Conference on Embedded Networked Sensor Systems (SenSys '06)* (2006), pp. 307–320.

[27] Bulusu, N., Bychkovskiy, V., Estrin, D., and Heidemann, J. Scalable, Ad Hoc Deployable, RF-Based Localization. In *Proc. of the Grace Hopper Celebration of Women in Computing Conference 2002* (Vancouver, Canada, 2002), pp. 1–5.

[28] Bulusu, N., Heideman, J., and Estrin, D. GPS-less Low Cost Outdoor Localization for Very Small Devices. *IEEE Personal Communications 7*, 5 (2000), 28–34.

[29] Bundesnetzagentur. Frequenzen für Ultra-Wideband-Technologie bereitgestellt. Pressemitteilung, 2008. `http://www.bundesnetzagentur.de/cae/servlet/contentblob/32372/publicationFile/1288/PM20080116UltraWidebandTechnologieId12430pdf.pdf` (Stand: 12.07.2011).

[30] Buschmann, C., Fekete, S. P., Fischer, S., Kröller, A., and Pfisterer, D. Koordinatenfreies Lokationsbewusstsein. *it - Information Technology, Themenheft Sensornetze 47*, 4 (2005), 70–87.

[31] Buschmann, C., Hellbrück, H., Fischer, S., Kröller, A., and Fekete, S. Radio Propagation-Aware Distance Estimation Based on Neighborhood Comparison. In *Proc. of the 4th European Conference on Wireless Sensor Networks (EWSN'07)* (Delft, The Netherlands, 2007), pp. 325–340.

[32] Buschmann, C., Krüger, D., and Fischer, S. Lean and Robust Phenomenon Boundary Approximation. In *Proc. of the 12th International Conference on Emerging Technologies and Factory Automation (ET-*

FA 07) (Patras, Greece, 2007), pp. 1202–1209.

[33] Buschmann, C., and Pfisterer, D. iSense: A modular hardware and software platform for wireless sensor networks. Tech. rep., 6. Fachgespräch "Drahtlose Sensornetze" der GI/ITG-Fachgruppe "Kommunikation und Verteilte Systeme", 2007.

[34] Buschmann, C., and Pfisterer, D. Potenziale und Grenzen spontan etablierter Lokationssysteme. *vfdb-Zeitung Forschung, Technik und Management im Brandschutz 3* (2008), 121–128.

[35] Butz, A., Baus, J., Krüger, A., and Lohse, M. A Hybrid Indoor Navigation System. In *Proc. of the 6th International Conference on Intelligent User Interfaces (IUI '01)* (Santa Fe, New Mexico, USA, 2001), pp. 25–32.

[36] Capkun, S., Hamdi, M., and Hubaux, J.-P. GPS-free Positioning in Mobile Ad Hoc Networks. In *Proc. of 34th Hawaii International Conference on System Sciences (HICSS '01)* (Maui, Hawaii, USA, 2001), pp. 157–167.

[37] coalesenses GmbH. HRM10 Demo Application. `http://www.coalesenses.com/index.php?page=hardware-docs-and-demos` (Stand: 12.07.2011).

[38] coalesenses GmbH. iSense Wireless Sensor Network Hardware Modules. `http://www.coalesenses.com/index.php?page=isense-hardware` (Stand: 12.07.2011).

[39] coalesenses GmbH. iShell User Guide. `http://www.coalesenses.com/index.php?page=development-environent` (Stand: 12.07.2011).

[40] Danckwardt, M., Pfisterer, D., and Fischer, S. WISEBED: Wireless Sensor Network Testbeds. In *Poster Session of the 14th Annual International Conference on Mobile Computing and Networking (MobiCom '08)* (San Francisco, California, USA, 2008).

[41] De Couto, D. S. J., Aguayo, D., Bicket, J., and Morris, R. A High-Throughput Path Metric for Multi-Hop Wireless Routing. *Wireless Networks 11* (2005), 419–434.

[42] de Paz Alberola, R., and Pesch, D. AvroraZ: Extending Avrora with an IEEE 802.15.4 compliant Radio Chip Model. In *Proc. of the 3rd Workshop on Performance Monitoring and Measurement of Heterogeneous Wireless and Wired Networks (PM^2HW^2N '08)* (Vancouver, Canada, 2008), pp. 43–50.

[43] Demirkol, I., Ersoy, C., and Alagoz, F. MAC Protocols for Wireless Sensor Networks: A Survey. *IEEE Communications Magazine 44*, 4 (2006), 115–121.

[44] Doherty, L., Pister, K., and Ghaoui, L. E. Convex Position Esti-

mation in Wireless Sensor Networks. In *Proc. of the 20th Annual Joint Conference of the IEEE Computer and Communications Societies (InfoCom '01)* (Anchorage, Alaska, USA, 2001), pp. 1655–1663.

[45] Dohler, M. Wireless Sensor Networks: The Biggest Cross-Community Design Exercise To-Date. *Bentham Recent Patents on Computer Science*, 1 (2008), 9–25.

[46] Dunkels, A. Full TCP/IP for 8-bit architectures. In *Proc. of the 1st International Conference on Mobile Systems, Applications and Services (MobiSys '03)* (San Francisco, California, USA, 2003), pp. 85–98.

[47] Dunkels, A., Grönvall, B., and Voigt, T. Contiki - A Lightweight and Flexible Operating System for Tiny Networked Sensors. In *Proc. of the 29th Annual Conference on Local Computer Networks (LCN '04)* (Tampa, Florida, USA, 2004), pp. 455–462.

[48] Durvy, M., Abeillé, J., Wetterwald, P., O'Flynn, C., Leverett, B., Gnoske, E., Vidales, M., Mulligan, G., Tsiftes, N., Finne, N., and Dunkels, A. Making Sensor Networks IPv6 ready. In *Proc. of the 6th International Conference on Embedded Network Sensor Systems (SenSys '08)* (Raleigh, North Carolina, USA, 2008), pp. 421–422.

[49] Elnahrawy, E., Francisco, J.-A., and Martin, R. P. Bayesian Localization in Wireless Networks using Angle of Arrival. In *Proc. of the 3rd International Conference on Embedded Networked Sensor Systems (SenSys '05)* (San Diego, California, USA, 2005), pp. 272–273.

[50] Elnahrawy, E., Li, X., and Martin, R. P. The Limits of Localization Using Signal Strength: A Comparative Study. In *The 1st Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON '04)* (Santa Clara, California, USA, 2004), pp. 406–414.

[51] EnOcean GmbH. Transmitter Module PTM 230 User Manual V1.12. `http://www.enocean.com/en/enocean_modules/PTM_230_User_Manual_V1.12.pdf` (Stand: 12.07.2011).

[52] Eriksson, J., Dunkels, A., Finne, N., Österlind, F., and Voigt, T. MSPsim – an Extensible Simulator for MSP430-equipped Sensor Boards. In *Proc. of the European Conference on Wireless Sensor Networks (EWSN '07), Poster/Demo session* (Delft, Netherlands, 2007).

[53] Eriksson, J., Österlind, F., Finne, N., Tsiftes, N., Dunkels, A., Voigt, T., Sauter, R., and Marrón, P. J. COOJA/MSPSim: Interoperability Testing for Wireless Sensor Networks. In *Proc. of the 2nd International Conference on Simulation Tools and Techniques (Simutools '09)* (Rome, Italy, 2009), pp. 1–7.

[54] Eswaran, A., Rowe, A., and Rajkumar, R. Nano-RK: An Energy-Aware Resource-Centric RTOS for Sensor Networks. In *Proc. of the 26th International Real-Time Systems Symposium (RTSS '05)* (Miami, Flori-

da, USA, 2005), pp. 256–265.

[55] Ettus Research LLC. Universal Software Radio Peripheral – Motherboard Data Sheet. http://www.ettus.com/downloads/ettus_ds_usrp_v7.pdf (Stand: 12.07.2011).

[56] Fekete, S. P., Kröller, A., Fischer, S., and Pfisterer, D. Shawn: The Fast, Highly Customizable Sensor Network Simulator. In *Proc. of the 4th International Conference on Networked Sensing Systems (INSS '07)* (2007), pp. 299–299.

[57] Fekete, S. P., Kröller, A., Pfisterer, D., Fischer, S., and Buschmann, C. Neighborhood-Based Topology Recognition in Sensor Networks. In *Proc. of the 1st International Workshop on Algorithmic Aspects of Wireless Sensor Networks (ALGOSENSORS '04)* (Turku, Finland, 2004), pp. 123–136.

[58] Fiala, M., and Basu, A. Robot Navigation using Panoramic Tracking. *Pattern Recognition 37*, 11 (2004), 2195–2215.

[59] Fischer, S., Zitterbart, M., and Buschmann, C. Projektbericht Sichere und flexible Grenz- und Liegenschaftsüberwachung durch drahtlose Sensornetze (FleGSens). Tech. rep., Universität Lübeck, Universität Karlsruhe, coalesenses GmbH, Germany, 2008.

[60] Fitzek, F. H. P., and Katz, M. D. *Cooperation in Wireless Networks: Principles and Applications – Real Egoistic Behavior is to Cooperate!* Springer, 2006.

[61] Fonseca, R., Gnawali, O., Jamieson, K., Kim, S., Levis, P., and Woo, A. The Collection Tree Protocol (CTP). http://www.tinyos.net/tinyos-2.x/doc/html/tep123.html (Stand: 12.07.2011).

[62] Free Software Foundation. eCos. http://ecos.sourceware.org/ (Stand: 12.07.2011).

[63] Free Software Foundation. GNU Radio. http://gnuradio.org/redmine/projects/gnuradio/wiki (Stand: 12.07.2011).

[64] Girod, L., and Estrin, D. Robust Range Estimation Using Acoustic And Multimodal Sensing. In *Proc. of the International Conference on Intelligent Robots and Systems (IROS '01)* (Maui, Hawaii, USA, 2001), pp. 1312–1320.

[65] Gnawali, O. The Link Estimation Exchange Protocol (LEEP). http://www.tinyos.net/tinyos-2.x/doc/html/tep124.html (Stand: 12.07.2011).

[66] Gnawali, O., Fonseca, R., Jamieson, K., Moss, D., and Levis, P. Collection Tree Protocol. In *Proc. of the 7th Conference on Embedded Networked Sensor Systems (SenSys'09)* (Berkeley, California, USA, 2009), pp. 1–14.

[67] Gura, N., Patel, A., Wander, A., Eberle, H., and Shantz, S. C. Comparing Elliptic Curve Cryptography and RSA on 8-bit CPUs. In *Cryptographic Hardware and Embedded Systems - CHES 2004*, M. Joye and J.-J. Quisquater, Eds., vol. 3156 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2004, pp. 925–943.

[68] Halkes, G., and Langendoen, K. Crankshaft: An Energy-Efficient MAC-Protocol for Dense Wireless Sensor Networks. In *Proc. of the 4th European Conference on Wireless Sensor Networks (EWSN '07)* (Delft, The Netherlands, 2007), pp. 228–244.

[69] Hansen, M. T. Asynchronous group key distribution on top of the cc2420 security mechanisms for sensor networks. In *Proc. of the 2nd ACM conference on Wireless network security (WiSec '09)* (Zurich, Switzerland, 2009), pp. 13–20.

[70] Harter, A., Hopper, A., Steggles, P., Ward, A., and Webster, P. The Anatomy of a Context-Aware Application. In *Proc. of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom '99)* (Seattle, Washington, USA, 1999), pp. 59–68.

[71] He, T., Huang, C., Blum, B. M., Stankovic, J. A., and Abdelzaher, T. Range-free Localization Schemes for Large Scale Sensor Networks. In *Proc. of the 9th Annual International Conference on Mobile Computing and Networking (MobiCom '03)* (San Diego, California, USA, 2003), pp. 81–95.

[72] Heinzelman, W. R., Chandrakasan, A., and Balakrishnan, H. Energy-efficient Communication Protocol for Wireless Microsensor Networks. In *Proc. of the 33rd Annual Hawaii International Conference on System Sciences (HICSS '00)* (Maui, Hawaii, USA, 2000), pp. 3005–3014.

[73] Heinzelman, W. R., Kulik, J., and Balakrishnan, H. Adaptive Protocols for Information Dissemination in Wireless Sensor Networks. In *Proc. of the 33rd Hawaii International Conference System Sciences (HISCC'00)* (Maui, Hawaii, USA, 2000), pp. 174–185.

[74] Hill, J. L., Szewczyk, R., Woo, A., Hollar, S., Culler, D. E., and Pister, K. S. J. System Architecture Directions for Networked Sensors. In *Proc. of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '00)* (Cambridge, Massachusetts, USA, 2000), pp. 93–104.

[75] Hofmann-Wellenhof, B., Lichtenegger, H., and Collins, J. *Global Positioning System: Theory and Practice*, 5 ed. Springer, 2001.

[76] Hokuyo Automatic Co., Ltd. Range-finder type laser scanner URG-04LX specifications. `http://www.acroname.com/robotics/parts/R283-HOKUYO-LASER1.pdf` (Stand: 12.07.2011).

[77] Hokuyo Automatic Co., Ltd. Laser Range Finder Overview,

2006. http://www.acroname.com/robotics/info/articles/laser/laser.html (Stand: 12.07.2011).

[78] Howard, A., Mataric, M. J., and Sukhatme, G. Relaxation on a Mesh: a Formalism for Generalized Localization. In *Proc. of the International Conference on Intelligent Robots and Systems (IROS 2001)* (Maui, Hawaii, USA, 2001), pp. 1055–1060.

[79] Hu, Y.-C., Perrig, A., and Johnson, D. Packet Leashes: a Defense against Wormhole Attacks in Wireless Networks. In *Proc. of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '03)* (San Franciso, California, USA, 2003), pp. 1976–1986.

[80] Hui, J. Deluge 2.0 - TinyOS Network Programming. http://www.cs.berkeley.edu/~jwhui/deluge/deluge-manual.pdf (Stand: 12.07.2011).

[81] Hui, J., and Culler, D. The dynamic behavior of a data dissemination protocol for network programming at scale. In *Proc. of the 2nd international Conference on Embedded Networked Sensor Systems (SenSys'04)* (Baltimore, Maryland, USA, 2004), pp. 81–94.

[82] IEEE 802.15 Working Group. IEEE 802.15 WPAN Task Group 4 (TG4). http://www.ieee802.org/15/pub/TG4.html (Stand: 12.07.2011).

[83] Iyengar, R., and Sikdar, B. Scalable and Distributed GPS free Positioning for Sensor Networks. In *Proc. of the International Conference on Communications (ICC '03)* (Anchorage, Alaska, USA, 2003), pp. 338–342.

[84] Jennic Ltd. Data Sheet: JN5139-001 and JN5139-Z01. http://www.jennic.com/files/product_briefs/JN-DS-JN5139-001-1v8.pdf (Stand: 12.07.2011).

[85] Jeong, J., and Culler, D. Incremental Network Programming for Wireless Sensors. In *Proc. of the 1st Annual Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON'04)* (Santa Clara, California, USA, 2004), pp. 25–33.

[86] Jinwala, D. C., Patel, D. R., and Dasgupta, K. S. Optimizing the block cipher resource overhead at the link layer security framework in the wireless sensor networks. *Lecture Notes in Engineering and Computer Science 2170*, 1 (2008), 770–775.

[87] Kahn, J. M., Katz, R. H., and Pister, K. S. J. Emerging challenges: Mobile networking for smart dust. *Journal of Communications and Networks 2*, 3 (2000), 188–196.

[88] Kargl, F. *Sicherheit in Mobilen Ad hoc Netzwerken.* PhD thesis, Fakultät für Informatik, Universität Ulm, Germany, 2003.

[89] Karlof, C., and Wagner, D. Secure Routing in Wireless Sensor Networks: Attacks and Countermeasures. *Ad Hoc Networks 1*, 2-3 (2003), 293–315.

[90] Karp, B., and Kung, H. T. GPSR: Greedy Perimeter Stateless Routing for Wireless Networks. In *Proc. of the 6th Annual International Conference on Mobile Computing and Networking (MobiCom '00)* (Boston, Massachusetts, USA, 2000), pp. 243–254.

[91] Kitasuka, T., Nakanishi, T., and Fukuda, A. Location Estimation System using Wireless Ad-Hoc Network. In *Proc. of the 5th International Symposium on Wireless Personal Multimedia Communications (WPMC '02)* (Honolulu, Hawaii, USA, 2002), pp. 305–309.

[92] Kröller, A., Fekete, S. P., Pfisterer, D., and Fischer, S. Deterministic Boundary Recognition and Topology Extraction for large Sensor Networks. In *Proc. of the 17th Annual Symposium on Discrete Algorithms (SODA '06)* (Miami, Florida, USA, 2006), pp. 1000–1009.

[93] Kröller, A., Pfisterer, D., Buschmann, C., Fekete, S. P., and Fischer, S. Shawn: A New Approach to Simulating Wireless Sensor Networks. In *Proc. of the 2005 Design, Analysis, and Simulation of Distributed Systems Symposium (DASD '05)* (San Diego, California, USA, 2005), pp. 117–124.

[94] Krüger, D., Pfisterer, D., and Buschmann, C. Efficient Selective Reprogramming in Sensor Networks, 2010. submitted for publication.

[95] Kuhn, F., Wattenhofer, R., Zhang, Y., and Zollinger, A. Geometric Ad-hoc Routing: of Theory and Practice. In *Proc. of the 22nd Annual Symposium on Principles of Distributed Computing (PODC '03)* (Boston, Massachusetts, USA, 2003), pp. 63–72.

[96] Landsiedel, O., Wehrle, K., and Götz, S. Accurate Prediction of Power Consumption in Sensor Networks. In *Proc. of the 2nd Workshop on Embedded Networked Sensors (EmNets '05)* (Sydney, Australia, 2005), pp. 37–44.

[97] Langendoen, K., and Reijers, N. Distributed Localization in Wireless Sensor Networks: A Quantitative Comparison. *Computer Networks 43*, 4 (2003), 499–518.

[98] Law, Y. W., Doumen, J., and Hartel, P. Survey and benchmark of block ciphers for wireless sensor networks. *ACM Transactions on Sensor Networks 2* (February 2006), 65–93.

[99] Levis, P., Clausen, T., Hui, J., Gnawali, O., and Ko, J. The Trickle Algorithm (RFC 6206). `http://tools.ietf.org/html/rfc6206` (March 2011).

[100] Levis, P., Lee, N., Welsh, M., and Culler, D. TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications. In *Proc. of*

the 1st International Conference on Embedded Networked Sensor Systems (SenSys '03) (Los Angeles, California, USA, 2003), pp. 126–137.

[101] Levis, P., Patel, N., Culler, D., and Shenker, S. Trickle: A Self-Regulating Algorithm for Code Propagation and Maintenance in Wireless Sensor Networks. In *Proc. of the 1st Symposium on Networked Systems Design and Implementation (NSDI '04)* (San Francisco, California, USA, 2004), pp. 15–28.

[102] Liu, A., and Ning, P. TinyECC: Elliptic Curve Cryptography for Sensor Networks (Version 2.0). `http://discovery.csc.ncsu.edu/software/TinyECC/` (Stand: 12.07.2011).

[103] Maroti, M., Völgyesi, P., Dora, S., Kusy, B., Nadas, A., Ledeczi, A., Balogh, G., and Molnar, K. Radio Interferometric Geolocation. In *Proc. of the 3rd International Conference on Embedded Networked Sensor Systems (SenSys '05)* (San Diego, California, USA, 2005), pp. 1–12.

[104] Mattern, F., and Römer, K. Drahtlose Sensornetze. *Informatik Spektrum 26*, 3 (2003), 191–194.

[105] Memsic. MCS410 Cricket Wireless Location System. `http://www.memsic.com/support/documentation/wireless-sensor-networks/category/7-datasheets.html?download=140%3Amcs410-cricket` (Stand: 12.07.2011).

[106] MEMSIC. MEMSIC: Sensor Boards. `http://www.memsic.com/products/wireless-sensor-networks/sensor-boards.html` (Stand: 12.07.2011).

[107] MEMSIC. MEMSIC: Wireless Modules. `http://www.memsic.com/products/wireless-sensor-networks/wireless-modules.html` (Stand: 12.07.2011).

[108] MEMSIC. MicaZ Data Sheet. `http://www.memsic.com/support/documentation/wireless-sensor-networks/category/7-datasheets.html?download=148` (Stand: 12.07.2011).

[109] MEMSIC. TelosB Data Sheet. `http://www.memsic.com/support/documentation/wireless-sensor-networks/category/7-datasheets.html?download=152` (Stand: 12.07.2011).

[110] Moses, R. L., Krishnamurthy, D., and Patterson, R. A Self-Localization Method for Wireless Sensor Networks. *Eurasip Journal on Applied Signal Processing, Special Issue on Sensor Networks 4* (2003), 358–358.

[111] Moses, R. L., and Patterson, R. M. Self-calibration of Sensor Networks. In *Proc. of SPIE Unattended Ground Sensor Technologies and Applications IV* (2002), pp. 108–119.

[112] Musaloiu-E., R., Liang, C.-J. M., and Terzis, A. Koala: Ultra-Low Power Data Retrieval in Wireless Sensor Networks. In *Proc. of the 7th International Conference on Information Processing in Sensor Networks (IPSN '08)* (St. Louis, Missouri, USA, 2008), pp. 421–432.

[113] Nagpal, R. Organizing a Global Coordinate System from Local Information on an Amorphous Computer. Memo 1666, Massachusetts Institute of Technology (MIT), A.I. Laboratory, Aug. 1999.

[114] Nagpal, R., Shrobe, H., and Bachrach, J. Organizing a Global Coordinate System from Local Information on an Ad Hoc Sensor Network. In *Proc. of the 2nd International Workshop on Information Processing in Sensor Networks (IPSN '03)* (Palo Alto, California, USA, 2003), pp. 333–348.

[115] Nasipuri, A., and Li, K. A Directionality based Location Discovery Scheme for Wireless Sensor Networks. In *Proc. of the 1st International Workshop on Wireless Sensor Networks and Applications (WSNA '02)* (Atlanta, Georgia, USA, 2002), pp. 105–111.

[116] National Institute of Standards and Technology. Skipjack and KEA Algorithm Specification (Version 2.0), May 1998. `http://csrc.nist.gov/groups/STM/cavp/documents/skipjack/skipjack.pdf` (Stand: 12.07.2011).

[117] Niculescu, D., and Nath, B. Ad Hoc Positioning System (APS). In *Proc. of the Global Telecommunications Conference (GLOBECOM '01)* (San Antonio, Texas, USA, 2001), pp. 2926–2931.

[118] Niculescu, D., and Nath, B. Ad Hoc Positioning System (APS) Using AoA. In *Proc. of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '03)* (San Franciso, California, USA, 2003), pp. 1734–1743.

[119] Niculescu, D., and Nath, B. Localized Positioning in Ad Hoc Networks. *Ad Hoc Networks 1*, 2-3 (2003), 247–259.

[120] Niculescu, D., and Nath, B. Position and Orientation in Ad Hoc Networks. *Ad Hoc Networks 2*, 2 (2004), 133–151.

[121] Niculescu, D., and Nath, B. VOR Base Stations for Indoor 802.11 Positioning. In *Proc. of the 10th Annual International Conference on Mobile Computing and Networking (MobiCom '04)* (Philadelphia, Pennsylvania, USA, 2004), pp. 58–69.

[122] Österlind, F., Dunkels, A., Eriksson, J., Finne, N., and Voigt, T. Cross-Level Sensor Network Simulation with COOJA. In *Proc. of the 1st International Workshop on Practical Issues in Building Sensor Network Applications (SenseApp '06)* (Tampa, Florida, USA, 2006), pp. 641–648.

[123] Patwari, N., and Hero III, A. O. Location Estimation Accuracy in

Wireless Sensor Networks. In *Proc. of the 36th Annual Asilomar Conference on Signals, Systems and Computers* (2002), pp. 1523–1527.

[124] Patwari, N., and Hero III, A. O. Using Proximity and Quantized RSS for Sensor Localization in Wireless Networks. In *Proc. of the 2nd International Workshop on Wireless Sensor Networks and Applications (WSNA '03)* (San Diego, California, USA, 2003), pp. 20–29.

[125] Patwari, N., O'Dea, R. J., and Wang, Y. Relative Location in Wireless Networks. In *Proc. of the 53rd Vehicular Technology Conference (VTC '01)* (Rhodes, Greece, 2001), pp. 1149–1153.

[126] Perrig, A., Stankovic, J., and Wagner, D. Security in Wireless Sensor Networks. *Communications of the ACM 47*, 6 (2004), 53–57.

[127] Pfisterer, D. *Comprehensive Development Support for Wireless Sensor Networks.* PhD thesis, Institute of Telematics, University of Luebeck, Germany, 2007.

[128] Pfisterer, D., and Buschmann, C. Coalescing Simulation and Embedded WSN Application Development. In *Proc. of the International Workshop on Sensor Network Engineering (IWSNE '08)* (Santorini, Greece, 2008).

[129] Pfisterer, D., Fischer, S., Kröller, A., and Fekete, S. P. Shawn: Ein alternativer Ansatz zur Simulation von Sensornetzwerken. Tech. rep., Institute for Pervasive Computing, ETH Zürich, 2005.

[130] Polastre, J., Hill, J., and Culler, D. Versatile Low Power Media Access for Wireless Sensor Networks. In *Proc. of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys '04)* (Baltimore, Maryland, USA, 2004), pp. 95–107.

[131] Priyantha, N. B., Chakraborty, A., and Balakrishnan, H. The Cricket Location-Support System. In *Proc. of the 6th Annual International Conference on Mobile Computing and Networking (MobiCom '00)* (Boston, Massachusetts, USA, 2000), pp. 32–43.

[132] Raykar, V. C., Kozintsev, I., and Lienhart, R. Position Calibration of Audio Sensors and Actuators in a Distributed Computing Platform. In *Proc. of the 11th International Conference on Multimedia (MULTIMEDIA '03)* (Berkeley, California, USA, 2003), pp. 572–581.

[133] Raykar, V. C., Kozintsev, I., and Lienhart, R. Self Localization of Acoustic Sensors and Actuators on Distributed Platforms. In *Proc. of the International Workshop on Multimedia Technologies in E-Learning and Collaboration (WOMTEC '03)* (Nice, France, 2003).

[134] Raymond, D. R., Marchany, R. C., Brownfield, M. I., and Midkiff, S. F. Effects of Denial-of-Sleep Attacks on Wireless Sensor Network MAC Protocols. *IEEE Transactions on Vehicular Technology 58*, 1 (2009), 367–380.

[135] Raymond, D. R., and Midkiff, S. F. Denial-of-Service in Wireless Sensor Networks: Attacks and Defenses. *Pervasive Computing 7*, 1 (2008), 74–81.

[136] Reddy, A. M. V., Kumar, A. P., Janakiram, D., and Kumar, G. A. Wireless Sensor Network Operating Systems - a Survey. *International Journal of Sensor Networks (IJSNET) 5*, 4 (2009), 236–255.

[137] Reijers, N., and Langendoen, K. Efficient Code Distribution in Wireless Sensor Networks. In *Proc. of the 2nd International Conference on Wireless Sensor Networks and Applications (WSNA '03)* (San Diego, California, USA, 2003), pp. 60–67.

[138] Robinson, D. P., and Marshall, I. W. An Iterative Approach to Locating Simple Devices in an ad-hoc Network. In *Proc. of the London Communications Symposium (LCS '03)* (London, England, 2003).

[139] Römer, K., and Mattern, F. The Design Space of Wireless Sensor Networks. *IEEE Wireless Communications 11*, 6 (2004), 54–61.

[140] Römer, K. The lighthouse location system for smart dust. In *ACM/USENIX Conference on Mobile Systems, Applications, and Services (MobiSys)* (San Francisco, CA, USA, 2003), pp. 15–30.

[141] Sallai, J., Balogh, G., Maróti, M., Lédeczi, Á., and Kusy, B. Acoustic Ranging in Resource-Constrained Sensor Networks. In *Proc. of the International Conference on Wireless Networks (ICWN '04)* (Las Vegas, Nevada, USA, 2004), pp. 467–473.

[142] Savarese, C., Rabaey, J., and Beutel, J. Locationing in Distributed Ad-Hoc Wireless Sensor Networks. In *Proc. of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP '01)* (Salt Lake City, Utah, USA, 2001), pp. 2037–2040.

[143] Savarese, C., Rabaey, J., and Langendoen, K. Robust Positioning Algorithms for Distributed Ad-Hoc Wireless Sensor Networks. In *Proc. of the USENIX Annual Technical Conference* (Monterey, California, USA, 2002), pp. 317–327.

[144] Savvides, A., Han, C.-C., and Strivastava, M. B. Dynamic Fine-Grained Localization in Ad-Hoc Networks of Sensors. In *Proc. of the 7th Annual International Conference on Mobile Computing and Networking (MobiCom '01)* (Rome, Italy, 2001), pp. 166–179.

[145] Savvides, A., Park, H., and Srivastava, M. B. The Bits and Flops of the N-Hop Multilateration Primitive for Node Localization Problems. In *Proc. of the 1st International Workshop on Wireless Sensor Networks and Application* (Atlanta, Georgia, USA, 2002), pp. 112–121.

[146] Savvides, A., Park, H., and Srivastava, M. B. The n-Hop Multilateration Primitive for Node Localization Problems. *Mobile Networks and Applications*, 8 (2003), 443–451.

[147] Seo, S. C., Han, D.-G., Kim, H. C., and Hong, S. TinyECCK: Efficient Elliptic Curve Cryptography Implementation over GF(2m) on 8-Bit Micaz Mote. *IEICE - Transactions on Information and Systems E91-D* (May 2008), 1338–1347.

[148] Seventh Framework Programme FP7 – Information and Communication Technologies (ICT). Wireless Sensor Networks Testbed Projekt (WISEBED). `http://www.wisebed.eu` (Stand: 12.07.2011).

[149] Seventh Framework Programme FP7 – Information and Communication Technologies (ICT). WiseML schema. `http://dutigw.st.ewi.tudelft.nl/wiseml/wiseml_schema.pdf` (Stand: 12.07.2011).

[150] Shah, R. C., and Rabaey, J. M. Energy Aware Routing for Low Energy Ad Hoc Sensor Networks. In *Proc. of the Wireless Communications and Networking Conference (WCNC '02)* (Orlando, Florida, USA, 2002), pp. 350–355.

[151] Shang, Y., Ruml, W., Zhang, Y., and Fromherz, M. P. J. Localization from Mere Connectivity. In *Proc. of 4th International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc '03)* (Annapolis, Maryland, USA, 2003), pp. 201–212.

[152] Simic, S. N., and Sastry, S. A Distributed Algorithm For Localization In Random Wireless Networks. *Discrete Applied Mathematics* (2003).

[153] Simic, S. N., and Sastry, S. Localization in Dense Random Sensor Networks. In *Proc. of the 3rd International Conference on Information Processing in Sensor Networks (IPSN '04)* (Berkeley, California, USA, 2004), pp. 1–6.

[154] Standards for Efficient Cryptography Group (SECG). SEC 2: Recommended Elliptic Curve Domain Parameters, January 2010.

[155] Stathopoulos, T., Heidemann, J., and Estrin, D. A Remote Code Update Mechanism for Wireless Sensor Networks. Tech. Rep. 30, Center for Embedded Networked Sensing (CENS), Los Angeles, CA, USA, 2003.

[156] Sundaram, N., and Ramanathan, P. Connectivity based Location Estimation Scheme for Wireless Ad hoc Networks. In *Proc. of the Global Telecommunications Conference (GLOBECOM '02)* (Taipei, Taiwan, 2002), pp. 143–147.

[157] T. Winter and P. Thubert and A. Brandt and T. Clausen and J. Hui and R. Kelsey and P. Levis and K. Pister and R. Struik and JP. Vasseur. RPL: IPv6 Routing Protocol for Low power and Lossy Networks. Draft, Internet Engineering Task Force, March 2011. `http://tools.ietf.org/html/draft-ietf-roll-rpl-19`.

[158] Takenga, C. M., and Kyamakya, K. Robust Positioning System based on Fingerprint Approach. In *Proc. of the 5th International Workshop on Mobility Management and Wireless Access (MobiWac '07)* (Chania,

Greece, 2007), pp. 1–8.

[159] Texas Instruments. CC2420 – 2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver (Rev. B). http://focus.ti.com/lit/ds/symlink/cc2420.pdf (Stand: 12.07.2011).

[160] Texas Instruments. MSP430F161x Data Sheet. http://focus.ti.com/lit/ds/slas368g/slas368g.pdf (Stand: 12.07.2011).

[161] Texas Instruments. MSP430x1xx User's Guide. http://focus.ti.com/lit/ug/slau049f/slau049f.pdf (Stand: 12.07.2011).

[162] The VINT Project. The Network Simulator – ns-2. http://www.isi.edu/nsnam/ns (Stand: 12.07.2011).

[163] TinyOS. http://www.tinyos.net/ (Stand: 12.07.2011).

[164] Titzer, B. L., Lee, D. K., and Palsberg, J. Avrora: Scalable Sensor Network Simulation with Precise Timing. In *Proc. of the 4th International Conference on Information Processing in Sensor Networks (IPSN '05)* (2005), pp. 477–482.

[165] Tsai, H.-W., Chu, C.-P., and Chen, T.-S. Mobile Object Tracking in Wireless Sensor Networks. *Computer Communications 30*, 8 (2007), 1811–1825.

[166] Tsiftes, N., Eriksson, J., and Dunkels, A. Low-Power Wireless IPv6 Routing with ContikiRPL. In *Proc. of the 9th International Conference on Information Processing in Sensor Networks (IPSN '10)* (Stockholm, Sweden, 2010), pp. 406–407.

[167] TU Delft. The MAC Alphabet Soup. http://www.st.ewi.tudelft.nl/~koen/MACsoup/protocols.php (Stand: 12.07.2011).

[168] TU Ilmenau. UKoLoS: UWB Communication. http://www-emt.tu-ilmenau.de/ukolos/uwb_communication.php (Stand: 12.07.2011).

[169] TU Ilmenau. UKoLoS: UWB Communication Applications. http://www-emt.tu-ilmenau.de/ukolos/apps_co.php (Stand: 12.07.2011).

[170] University of Bonn. BonnMotion Documentation. http://net.cs.uni-bonn.de/fileadmin/ag/martini/projekte/BonnMotion/src/BonnMotion_Docu.pdf (Stand: 12.07.2011).

[171] van Dam, T., and Langendoen, K. An Adaptive Energy-Efficient MAC Protocol for Wireless Sensor Networks. In *Proc. of the 1st International Conference on Embedded Networked Sensor Systems (SenSys '03)* (Los Angeles, California, USA, 2003), pp. 171–180.

[172] Wang, Q., Zhu, Y., and Cheng, L. Reprogramming Wireless Sensor Networks: Challenges and Approaches. *IEEE Network 20*, 3 (2006), 48–55.

[173] Want, R., Hopper, A., Falcao, V., and Gibbons, J. The Active

Badge Location System. *ACM Transactions on Information Systems 10*, 1 (1992), 91–102.

[174] Warneke, B., Last, M., Liebowitz, B., and Pister, K. S. J. Smart dust: Communicating with a cubic-millimeter computer. *Computer 34*, 1 (2001), 44–51.

[175] Whitehouse, C. D. The Design of Calamari: an Ad-hoc Localization System for Sensor Networks. Master's thesis, University of California, Berkeley (UCB), 2002.

[176] Whitehouse, K., and Culler, D. Calibration as Parameter Estimation in Sensor Networks. In *Proc. of the 1st International Workshop on Wireless Sensor Networks and Applications (WSNA '02)* (Atlanta, Georgia, USA, 2002), pp. 59–67.

[177] Whitehouse, K., Karlof, C., Woo, A., Jiang, F., and Culler, D. The Effects of Ranging Noise on Multihop Localization: An Empirical Study. In *Proc. of the 4th International Conference on Information Processing in Sensor Networks (IPSN '05)* (Los Angeles, California, USA, 2005), pp. 73–80.

[178] Wireshark Foundation. Wireshark – The world's foremost network protocol analyzer. `http://www.wireshark.org` (Stand: 12.07.2011).

[179] Wood, A. D., and Stankovic, J. A. Denial of Service in Sensor Networks. *Computer 35*, 10 (2002), 54–62.

[180] Xu, W., Trappe, W., Zhang, Y., and Wood, T. The Feasibility of Launching and Detecting Jamming Attacks in Wireless Networks. In *Proc. of the 6th International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc '05)* (Urbana-Champaign, Illinois, USA, 2005), pp. 46–57.

[181] Ye, W., Heidemann, J., and Estrin, D. An Energy-Efficient MAC Protocol for Wireless Sensor Networks. In *Proc. of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '02)* (New York, USA, 2002), pp. 1567–1576.