Institut für Parallele und Verteilte Systeme
Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Bachelorarbeit Nr. 1

Effiziente Strategien zur Aufgabenverteilung für Public Sensing Systeme

Steffen Maaß

Studiengang:	Informatik
Prüfer:	Prof. Dr. rer. nat. Dr. h. c. Kurt Rothermel
Betreuer:	M. Sc. Patrick Baier
begonnen am:	15. März 2012
beendet am:	14. September 2012
CR-Klassifikation:	C.2.1, C.2.4, C.3

Kurzfassung

Die vorliegende Bachelorarbeit stellt für die Aufgabenverteilung im Bereich des Public Sensing verschiedene Optimierungsansätze vor. Public Sensing ist dabei eine neue Entwicklung im Bereich der Sensornetzwerke, bei der anstelle von stationären Sensorbojen normale Smartphones eingesetzt werden. Public Sensing stellt damit eine kostengünstige Alternative zu herkömmlichen Sensornetzwerken dar.
Ein Problem des Public Sensings liegt im hohen Energieaufwand für die Endgeräte der Teilnehmer, daher sollen in dieser Arbeit effizientere Strategien für einzelne Komponenten des Public Sensing-Systems vorgestellt und evaluiert werden. Der Fokus liegt dabei zum einen auf einer effizienteren Form der Positionserfassung, wobei bei dem hier vorgestellten Vorgehen die Anzahl der Positionserfassungen verringert werden soll. Zum anderen soll für die Verteilung der Aufgaben an die Teilnehmer eine verbesserte Strategie vorgestellt werden, wobei diese Strategie auf einer adaptiven Verteilung der Aufgaben beruht. Die Teilnehmer erhalten dabei nur die Aufgaben, die sie auch potentiell erfüllen können. Dazu wird zuerst ein einfacheres Vorgehen auf Basis der euklidischen Distanzen vorgestellt. Der Vorteil dieses Vorgehens liegt darin, dass weniger Ansprüche an das System gestellt werden und diese Strategie damit universeller verwendet werden kann. Darauf aufbauend soll ein Ansatz vorgestellt werden, der zusätzlich noch die möglichen Wege der Teilnehmer mit berechnet und damit weitere Effizienzsteigerungen erlaubt, wobei sich bei diesem Ansatz die Teilnehmer nur auf den Wegen einer festgelegten Karte bewegen dürfen.
Die verschiedenen Optimierungen werden schlussendlich untereinander und gegenüber dem Standard-System verglichen, um die Verbesserungen der Effizienz der entwickelten Ansätze zu bestätigen.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Verzeichnis der Algorithmen

1 Einleitung

1.1 Motivation

Die heutzutage allgegenwärtige Präsenz von Smartphones, welche mit einer hochwertigen Ausstattung an internen Sensoren aufwarten können, macht die gemeinschaftliche Messung verschiedenster Umgebungsparameter wie beispielsweise den Geräuschpegel möglich. Die Verwendung von Smartphones im Bereich der Entwicklung und des Aufbaus von Sensornetzwerken ist dazu ein neues Denkmuster, welches im Folgenden als „Public Sensing“ deklariert werden soll. Durch die Verwendung der Smartphones anstelle von teuren Sensorbojen stellt das Public Sensing eine kostengünstige Alternative im Vergleich zu herkömmlichen Sensornetzwerken dar. Die Idee des Public Sensings liegt dabei darin, dass Smartphones Umgebungsdaten wie beispielsweise die Lautstärke erfassen und an eine zentrale Instanz melden, wobei sich die Smartphones mit dem jeweiligen Benutzer in einem bestimmten Gebiet bewegen, für welches die jeweiligen Daten erhoben werden sollen.
Die Verwendung von Smartphones ist dabei besonders für die Zwecke des Public Sensings geeignet, da diese Geräte über den reinen Zweck der mobilen Kommunikation hinaus oft viele zusätzliche Fähigkeiten besitzen, wie zum Beispiel Sensoren zur Bestimmung der aktuellen Lautstärke, der aktuellen Umgebungshelligkeit sowie einen GPS-Empfänger zur Bestimmung der aktuellen Position. Aufgrund der genannten guten Ausstattung und der hohen Mobilität der Benutzer ergibt sich die Möglichkeit, mittels des Public Sensing ein statisches Sensornetzwerk zu ersetzen und möglicherweise auch eine verbesserte Datenlage zu erzielen.

Allerdings müssen für eine hohe Akzeptanz des Public Sensings mehrere Probleme gelöst werden: Ein potentiell großer Nachteil für Teilnehmer des Public Sensings ist der hohe Energieverbrauch, da zum Datenaustausch eine Datenverbindung bestehen muss (UMTS, WLAN, etc.). Zudem benötigt das Aktivieren der eingebauten Sensoren einen nicht unerheblichen Anteil der stark begrenzen Energievorräte des Smartphones. Darüber hinaus stellt die Positionsbestimmung per GPS ein weiteres Problem hinsichtlich des Energieaufwands dar.
Der hohe Energieaufwand des Public Sensings und dessen Verbesserung wird umso wichtiger, wenn die Motivation der Teilnehmer des Public Sensings miteinbezogen wird: Die Teilnehmer erhalten in den meisten Fällen für ihre Teilnahme keine Vergütung, einzig das gemeinschaftliche Interesse an den gesammelten Daten lässt sie am Public Sensing teilnehmen, wobei umso bessere Ergebnisse erzielt werden, je mehr Teilnehmer sich finden. Daher ist es besonders wichtig, den Ressourcenverbrauch so gering wie möglich zu halten, um bei den Teilnehmern auf eine hohe Akzeptanz zu stoßen.

Aufgrund dieser Anforderungen liegt der Fokus aktueller Arbeiten darauf, den Energieaufwand für das gesamte Public Sensing-System zu minimieren, wobei durch geeignete Ansätze die Effektivität des Systems erhalten werden soll. Aus den oben genannten Gründen, liegt der Schwerpunkt hierbei vor allem auf der Minimierung des Ressourcenaufwands für den einzelnen Teilnehmer.

In dieser Arbeit werden Strategien zur Aufgabenverteilung in Public Sensing-Systemen vorgestellt, deren Ziel eine Effizienzsteigerung des gesamten Systems darstellt, wobei die Effektivität im Vergleich zum naiven Ansatz erhalten bleiben soll.
Ein Optimierungsziel ist dabei die Minimierung der Anzahl der Aktivierungen und Verwendungen des Positionserfassungssystems der Teilnehmer, ein anderes die Minimierung der Teilnehmer verschicken Aufgaben. Das zentrale Ziel ist dabei die Minimierung des Energieaufwands des Gesamtsystems, wobei dies meist gleichbedeutend mit der Minimierung des Energieaufwands für teilnehmende mobile Knoten ist.

1.2 Aufbau

Die vorliegende Arbeit gliedert sich in insgesamt sieben Kapitel inklusive dieser Einleitung und einer Zusammenfassung.

Aufbauend auf dieser Einleitung soll in Kapitel 2 das verwendete Systemmodell vorgestellt werden, welches die Grundlage für die später beschriebenen Ansätze definiert. Kapitel 3 stellt verwandte Arbeiten in dem Forschungsbereich „Public Sensing“ vor und grenzt die vorliegende Arbeit von anderen Arbeiten ab. Kapitel 4 beschreibt detailliert den Entwurf der Kernkomponenten, wobei auf die Schlüsselprobleme des Systems eingegangen wird. In Kapitel 5 wird die Implementierung beschrieben, wobei auch die Konzepte des Entwurfskapitels genauer hinsichtlich der effizienten Implementierung beschrieben werden. Die Evaluation der vorgestellten Ansätze und der daraus gewonnenen Ergebnisse erfolgt in Kapitel 6, Kapitel 7 beschließt die Arbeit mit einem Ausblick auf potentielle weitere Verbesserungen und einer Zusammenfassung.

2 Systemmodell

Das folgende Kapitel soll über das verwendete Systemmodell der vorliegenden Arbeit informieren. Dafür sollen die Bestandteile des Public Sensing-Systems vorgestellt und die Anforderungen an Diese formuliert werden:

Das System besteht aus einer Menge von mobilen Knoten (Smartphones oder ähnliche Geräte, die im Zuge des Public Sensings ihre Sensoren zur Verfügung stellen), die über die Möglichkeit der Positionserfassung verfügen sowie eine Menge an Sensoren zur Verfügung stellen. Diese Knoten müssen zudem über die Möglichkeit zur Kommunikation mit einem Server verfügen, dieser dient zum einen als zentrale Verarbeitungseinheit für eintreffende Daten der mobilen Knoten und zum anderen als Interface für den Benutzer, der über den Server Aufgaben definieren und gesammelte Daten einsehen kann.

Die Positionserfassung der mobilen Knoten muss dabei bis auf einen begrenzten Fehler genau sein, um die Effektivität des Systems zu gewährleisten, die mobilen Knoten dürfen sich zudem nur auf vordefinierten Wegen bewegen, wobei diese Wege beispielsweise durch einen Kartengraph definiert werden. Der Kommunikationskanal von mobilen Knoten zum Server muss zudem zu jeder Zeit bestehen um, ein effektives System zu gewährleisten.

Grafik 2.1 stellt einen Überblick über die am Public Sensing-System beteiligten Komponenten dar.

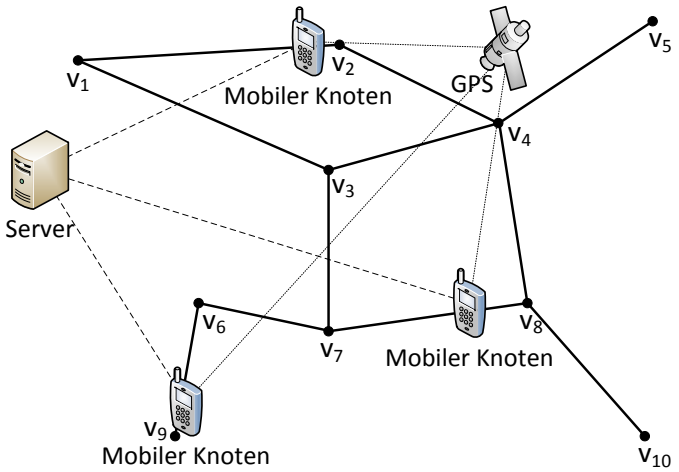


Abbildung 2.1: Grafischer Überblick über die am Public Sensing-System beteiligten Komponenten

2.1 Energie

Zusätzlich zu den oben beschriebenen Komponenten sollen nun einzelne Teilbereiche des Systems genauer charakterisiert werden, beginnend mit dem Energiemodell für das Kommunikationssystem gefolgt von einem weiteren Energiemodell für das Positionserfassungssystem, welches durch das Global Positioning System (GPS) realisiert wird.

2.1.1 Kommunikation

Da die mobilen Knoten auch in Umgebungen ohne Festnetzanbindung operieren sollen, basiert das Kommunikationssystem auf einem Funknetzwerk, wobei dieses im Folgenden nur für die Berechnung der aufgewandten Energie spezifiziert wird. Die Ermittlung des Energieaufwands für das Kommunikationssystem ermöglicht eine genauere Analyse und motiviert eine Evaluation hinausgehend über die reine Anzahl gesendeter Nachrichten hin zum Vergleich des Energieaufwands unterschiedlicher Ausgangssituationen.

Zur Berechnung des Energieaufwands einer einzelnen Nachricht, bzw. einer Sequenz von Nachrichten wurde die Analyse aus [BBV09] verwendet. Der Energieaufwand für eine Nachricht bzw. für eine Sequenz an Nachrichten setzt sich dabei aus mehreren Einzelkomponenten zusammen:

- die Energie, die für den Aufbau der Verbindung aufgewandt wird
- die Energie die für den Versand einzelner Datenpakete aufgewandt wird
- die Energie für das kurzfristige Aufrechterhalten der Verbindung nach dem Versenden oder Empfangen des letzten Datenpakets

Dabei wird, wie im 3GPP-Standard [3gp] und in [LYC04] beschrieben, der Zustand der Netzwerkverbindung für kurze Zeit in einem Zustand mit höherem Energieaufwand belassen, in welchem dem Gerät ein eigener Kanal mit der Basisstation zugesichert ist, wobei dieser eine geringe Verzögerung und einen hohen Durchsatz aufweist. Dieses Vorgehen wird angewandt, da nicht klar ist, ob nach dem Versenden oder Empfangen eines Datenpaketes weitere Pakete folgen oder die Übertragung beendet wird, aber ein sofortiges Umschalten in den Leerlauf-Zustand bei weiteren Datenpaketen einen sehr großen Energieaufwand bedeuten würde, daher wird der Zustand mit hohem Energieaufwand noch einige weitere Sekunden nach der letzten Übertragung aktiv gehalten.
Die Tabelle 2.1 gibt dabei einen Überblick über die Ergebnisse der Messungen aus [BBV09], auf welche das darauf folgende Energiemodell aufbaut.

Damit kann nun das Energiemodell definiert werden, die Einzelkomponenten definieren sich dabei wie folgt:

- $R(x)$ bezeichnet dabei den Energieaufwand für den Verbindungsaufbau und die Übertragung von x Kilobyte (kB) an Daten

Energieaufwand für den Verbindungsaufbau	3.5 J
Energieaufwand pro übertragenes kB	0.025 J
Energie für die Aufrechterhaltung der Verbindung nach letzter Übertragung	0.62 W
Länge der Aufrechterhaltung der Verbindung nach letzter Übertragung	12.5 s

Tabelle 2.1: Aufzählung der verschiedenen Phasen mit Werten zur Berechnung des Energieaufwands einer 3G-Verbindung, Werte aus [BBV09]

- $T(t_{offset})$ bezeichnet den Energieaufwand für das Aufrechterhalten der Verbindung mit t_{offset} als Abstand in Sekunden seit der vorangegangenen Datenübertragung
- $G(x, t_{offset})$ bezeichnet den Gesamtaufwand für x kB Daten mit einem Abstand von t_{offset} Sekunden seit der vorangegangenen Verbindung

Die Formel (2.1) fasst die Einzelkomponenten nun zusammen:

$$\begin{aligned} T(t_{offset}) &= 0.62\ W * \min(12.5\ s, t_{offset}) \\ R(x) &= 3.5\ J + 0.025\ J * x \\ G(x, t_{offset}) &= R(x) + T(t_{offset}) \end{aligned} \tag{2.1}$$

Damit kann die Gesamtenergie $e(tr(x_0, t_0), tr(x_1, t_1), \ldots, tr(x_n, t_n))$ für n aufeinanderfolgende Einzelübertragungen definiert werden mit $tr(x_i, t_i)$ als Energieaufwand für eine Übertragung mit der Datenmenge x_i kB zum Zeitpunkt t_i, Formel (2.2) zeigt die Berechnung von $e(\ldots)$ auf. Der Term $G(x_n, \infty)$ entsteht dadurch, dass nach der letzten Datenübertragung keine weitere Übertragung vorgenommen wird, der Zeitabstand zur nächsten Übertragung also $\infty\ s$ beträgt.

$$e(tr(x_0, t_0), tr(x_1, t_1), \ldots, tr(x_n, t_n)) = \sum_{i=0}^{n-1} (G(x_i, (t_{i+1} - t_i))) + G(x_n, \infty) \tag{2.2}$$

2.1.2 Global Positioning System

Ein weiterer Aspekt bei der Betrachtung des Energieaufwands des Public Sensing-Systems ist die zur Positionserfassung aufgewandte Energie. Daher soll nun ein Energiemodell für das Global Positioning System eingeführt werden: Die folgenden Zahlen beruhen dabei auf Beobachtungen und Messungen die im Rahmen des EnTracked-Projekts [KLGT09] durchgeführt wurden.
Das Energiemodell baut dabei auf den Verzögerungen bei der Positionserfassung und der daraus hergeleiteten Formel (2.4) auf.
Zur Analyse des Energieaufwands wird im weiteren Verlauf die folgende Notation verwendet: $p(t_i)$ symbolisiert eine GPS-Positionserfassung zum Zeitpunkt t_i, $e(p(t_i), p(t_{i+1}, \ldots, p(t_j)))$

entspricht der Energie, die für die aufeinanderfolgenden Positionserfassungen $p(t_i)$ bis $p(t_j)$ verwendet wurde.
Aus [KLGT09] wird der durchschnittliche Energieaufwand für ein aktiviertes GPS von 0.324 W übernommen, damit ergibt sich für eine Folge von n aufeinanderfolgenden GPS-Positionserfassungen die Formel (2.3).

$$e(p(t_0), p(t_1), \ldots, p(t_n)) = \sum_{i=0}^{n-1} (\min(t_{i+1} - t_i, 36\ s) * 0.324\ W) + 36\ s * 0.324\ W \tag{2.3}$$

Dabei wird die für die letzte Positionserfassung aufgebrachte Energie nicht miteinbezogen, da diese den Startwert für die nächste Sequenz vorgibt. Der konstante Term 36 s ergibt sich aus den Beobachtungen in [KLGT09], welche für das GPS eine Initialisierungsphase vor der ersten Positionserfassung und eine Aktivitätsphase nach der letzten Positionserfassung beobachten konnten. Der Grund für die Initialisierungsphase ist dabei leicht ersichtlich: Nach einer längeren Deaktivierungsphase benötigt das GPS eine Kalibrierungsphase mit den aktuell sichtbaren Satelliten, die Aktivitätsphase nach einer Positionserfassung lässt sich damit erklären, dass hierbei durch das GPS versucht wird, das Nutzerverhalten abzuschätzen und weitere Positionserfassungen, die Initiale folgen, günstiger und ohne erneute Initialisierungsphase zu gestalten. Für die Initialisierungsphase des GPS wurden experimentell 6 s, für die Aktivitätsphase nach einer Positionserfassung 30 s ermittelt.

2.2 Positionserfassung

Das Systemmodell fordert die Notwendigkeit, dass es den mobilen Knoten zeitnah möglich sein muss, ihre aktuelle Position zu bestimmen. Dies wird in der vorliegenden Arbeit durch eine Simulation von GPS erreicht. Im folgenden Abschnitt soll daher das verwendete Modell des GPS charakterisiert werden.

2.2.1 Positionserfassungsungenauigkeit

Mithilfe des GPS kann ein mobiler Knoten seine Position nur bis auf einen Fehler ϵ genau bestimmen, dieser Fehler resultiert dabei aus verschiedenen Quellen wie atmosphärischen Einflüssen, minimalen Orbit-Veränderungen der Satelliten, Rundungsfehlern und minimalen Zeitunterschieden zwischen den Satelliten und dem Empfangsgerät [PKG10].

Da ein Teil der vorgestellten Arbeit auf Simulationen beruht, soll eine Möglichkeit gefunden werden, eine Approximation des realen GPS-Verhaltens in Bezug auf die Ungenauigkeit der Positionserfassung zu erreichen.
Dabei ist im Zuge der Simulation ist die aktuelle, genaue Position $pos_{genau} = (x, y)$ eines mobilen Knotens gegeben, diese wird dann mittels der folgenden Umrechnung und der

Verwendung der beiden Zufallszahlen $radius \in [0,1)$ und $winkel \in [0,2\pi)$ zur aktuellen, bis auf ϵ genauen Position $pos_{GPS} = (x', y')$ transformiert:

$$pos_{GPS}.x' = pos_{genau}.x + radius * \epsilon * \cos(winkel)$$
$$pos_{GPS}.y' = pos_{genau}.y + radius * \epsilon * \sin(winkel)$$

Die mithilfe dieser Berechnung entstehende Verteilung ist in Grafik 2.2 dargestellt, wobei die quadratische Verteilung der einzelnen Positionen in Abhängigkeit vom Mittelpunkt klar wird.

Der Fehler ϵ wird dabei im Folgenden als konstant angenommen mit einem Wert von 15 m, dies ist ein in der Realität mit solchen GPS-Empfängern, wie sie in Smartphones verwendet werden, und in einer urbanen Umgebung durchaus erreichter Wert, wenn mindestens 4 Satelliten in Sicht sind [PKG10].

2.2.2 Verzögerte Positionsbestimmung

Zusätzlich zur Ungenauigkeit der Positionserfassung wird eine weitere Eigenschaft des GPS simuliert, die Verzögerung beim Erhalt neuere Positionsinformationen: Diese ist dabei abhängig von der Inaktivitätsdauer des GPS vor der aktuellen Anfrage. Dies hat seinen Ursprung darin, dass in Realität der GPS-Chip nach einer Positionserfassung nicht sofort wieder abgeschaltet wird, sondern eine gewisse Zeit, beispielsweise 30 s, weiter aktiviert bleibt, um nachfolgende Positionserfassungen schneller durchführen zu können. Innerhalb dieser Aktivitätsphase können weitere Positionserfassungen schnell durchgeführt werden, für diese Arbeit wurde für schnelle Positionserfassungen, bzw. Positionserfassungen innerhalb von 30 s seit der letzten Positionserfassung, ein Verzögerungswert von 1 s angenommen.

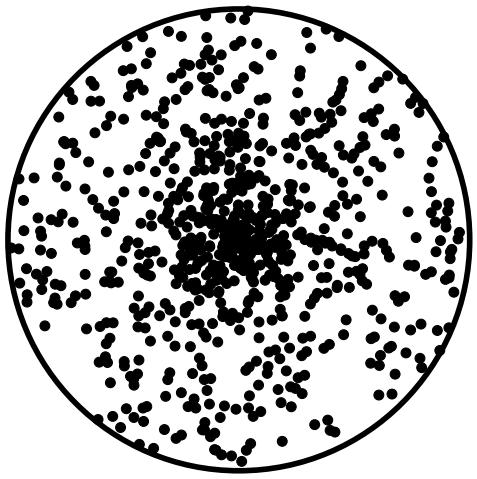


Abbildung 2.2: Beispielhafte Verteilung der Positionserfassungsungenauigkeit bei GPS-Positionserfassungen mit 800 erfassten Positionen, die Ausgangsposition befindet sich im Mittelpunkt des Kreises

Muss dagegen das GPS erst aktiviert werden so wurde ein Verzögerungswert von 6 s angenommen, die Formel (2.4) fasst dies zusammen.

$$delay(t_{offset}) = \begin{cases} 1\ s & t_{offset} \leq 30 \\ 6\ s & t_{offset} > 30 \end{cases} \tag{2.4}$$

wobei t_{offset} die Zeit seit der letzten Positionserfassung mithilfe des GPS symbolisiert. Diese Daten wurden dem EnTracked-Projekt [KLGT09] entnommen, wobei die Daten experimentell durch Messungen am Beispiel eines Nokia N95-Telefons gewonnen wurden.

3 Verwandte Arbeiten

Dieses Kapitel soll einen Überblick über andere Arbeiten im Themenfeld des Public Sensings geben:

Der Begriff Public Sensing bezeichnet das kollaborative Sammeln von Umweltdaten, meist unter Zuhilfenahme von Smartphones, wobei in den letzten Jahren auch andere Begriffe für dieses Vorgehen publiziert wurden wie: „People-Centric Urban Sensing“ [CEL+06], „Urban Sensing“ [CHK08], „Participatory Sensing“ [BEH+06], „Mobile Phone Sensing“ [LML+10] oder „Collaborative Sensing“ [CKK+08].
Dazu wurde in [CEL+08] der Begriff „Public Sensing“ als Teilmenge des „People-Centric Sensing“ definiert, wobei mittels des „Public Sensing“ im Gegensatz zum „Personal Sensing“ oder „Social Sensing“ zum öffentlichen Wohl beigetragen wird.

Im Bereich des Public Sensings wurden in den letzten Jahren mehrere Arbeit veröffentlicht, mit dem Ziel herkömmliche Sensornetzwerke zu ersetzen. Im Zuge dieser Bemühungen wurden einige Systeme zu diesem Zweck entwickelt wie MobGeoSen [KBP+08], CenceMe [MLEC07], Kohlenstoffmonooxid-Messungen [SM08] oder zur Überwachung der Lärmbelästigung [MSN+09], wobei all diese Systeme den Zweck der gesammelten Daten auf ein Themengebiet reduzieren.
Im Gegensatz dazu wurden Systeme entworfen wie [PDR11] oder [BDR12], welche den Zweck der gesammelten Daten offen lassen und das Erfassen der Daten in den Vordergrund stellen, das in dieser Arbeit vorgestellte System ist dabei in dieselbe Kategorie einzuordnen, wobei das in dieser Arbeit vorgestellte System die effiziente Aufgabenverteilung in den Vordergrund stellt.

In anderen Ansätzen wie in [RES10], worin die Nutzer nach ihrer Verlässlichkeit hinsichtlich der Erfüllung der Aufgabe ausgesucht werden, oder [RB12], bei dem mithilfe von ortsabhängigen Spielen die Teilnehmer an bestimmte Orte geführt werden sollen, wird eine Kontrolle über die Mobilität der teilnehmenden Nutzer mit bei der Verteilung der Aufgaben einbezogen. Im Gegensatz dazu soll diese Arbeit ein System vorstellen, welches ohne die Möglichkeit der Kontrolle über die Mobilität der Nutzer funktioniert.

Ein weiteres Arbeitsfeld im Bereich des Public Sensing liegt im Datenschutz für die Teilnehmer des Public Sensings: Innerhalb eines Public Sensing-Systems sind typischerweise viele Attribute der einzelnen Teilnehmer sichtbar, was zu einem relativ großen Grad an Überwachung führt. Auch diesem Aspekt des Public Sensings wird in verschiedenen Projekten Rechnung getragen wie beispielsweise in „Anonysense“ [CKK+08] oder „PrivaSense“ [OLFM07].

Für den Bereich der drahtlosen Sensornetzwerken existieren bereits Algorithmen wie in [SP01] zur Maximierung der Abdeckung eines Bereichs durch statische platzierte Sensoren, allerdings lassen sich diese nicht auf die in dieser Arbeit vorliegende Problemstellung übertragen, da anders als in einem statischen Sensornetz in dem hier behandelten Public Sensing-System keine harten Garantien vergeben werden können, ob und wann eine Erfassung eines Sensorwertes möglich ist.

Für das Public Sensing finden sich einige direkte Anwendungen, wie beispielsweise die Beobachtung der Lärmbelästigung in [SM08] und [Kan10]. Ein anderes Beispiel zur Anwendung findet sich in der automatisierten Validierung von Straßenkarten, die gemeinschaftlich für Kartendienste wie OpenStreetMap erstellt werden [BWDR11].

4 Entwurf

Das folgende Kapitel führt in die Konzeption des grundlegenden Systems und der darauf aufbauenden, verbesserten Ansätze zur Aufgabenverteilung ein. Darüber hinaus soll auch für die Positionserfassung ein verbesserter Ansatz eingeführt werden.

4.1 Grundlegendes System

Im diesem Abschnitt wird das Basis-System beschrieben, auf welches verbesserte Ansätze aufbauen und welches im Zuge dieser Arbeit schrittweise verbessert werden soll. Das Basis-System oder naive System kann dabei so genannte Benutzeranfragen verarbeiten und ausführen.

Diese Anfragen bestehen dabei aus einer Menge an Orten, die jeweils mithilfe eines Radius um den jeweiligen Ort eine Erfassungszone definieren, des Weiteren definiert jeder Ort eine Menge von Sensoren, die an diesem Ort erfasst werden sollen. Ein einzelner Ort mit den genannten Attributen wird dabei virtueller Sensor genannt. Grafik 4.1 visualisiert eine Menge an virtuellen Sensoren $S = \{s_1, s_2, s_3, s_4\}$ die auf einem Gebiet mit Graph definiert wurden, wobei die Menge S damit eine Benutzeranfrage darstellt. Jede Benutzeranfrage wird dabei auch als Aufgabe bezeichnet.

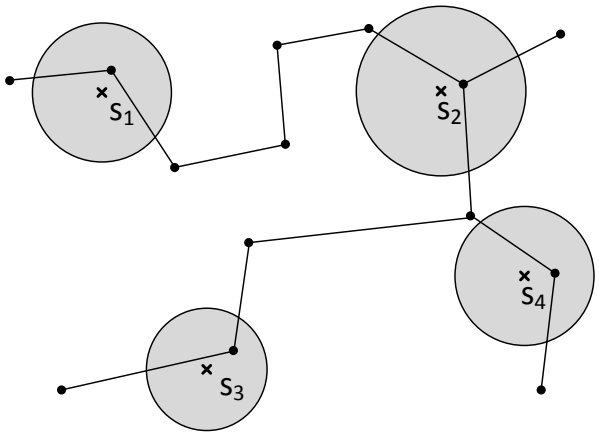


Abbildung 4.1: Gebiet mit vier definierten virtuellen Sensoren, die graue Kreisfläche um den mit einem Kreuz bezeichneten Mittelpunkt visualisiert die Erfassungszone des jeweiligen virtuellen Sensors

Die Ausführung dieser Anfragen hat dabei den folgenden Ablauf: Am Public Sensing-System teilnehmende mobile Knoten erhalten alle aktuell vom Benutzer angefragten und noch nicht erfassten virtuellen Sensoren. Wird von einem mobilen Knoten durch eine Positionserfassung und dem Abgleich der dabei erhaltenen Position mit den Daten der virtuellen Sensoren erkannt, dass der betreffende mobile Knoten sich innerhalb der Erfassungszone eines virtuellen Sensors befindet, so sendet der mobile Knoten eine Nachricht an den Server, welche die aktuelle Position, den soeben erfasste virtuellen Sensor und die vom entsprechenden virtuellen Sensor verlangten Sensordaten beinhaltet.

Als formale Grundlage soll nun das Konzept des virtuellen Sensors definiert werden:

Ein virtueller Sensor hat die folgende Attribute: Die Position p, den Radius r um die entsprechende Position herum innerhalb dessen ein virtueller Sensor erfasst werden kann, die Anzahl an Messungen k die an dem entsprechenden virtuellen Sensor von paarweise verschiedenen mobilen Knoten vorgenommen werden müssen, damit dieser als erfasst gilt, eine Zeitmarke t_0 welche die früheste Auslieferungszeit des virtuellen Sensors kennzeichnet, eine Zeitmarke t_1 bis wann der entsprechende Sensor spätestens erfasst werden muss, um nicht als verpasst zu gelten, und eine Menge T an zu erfassenden Sensortypen (zum Beispiel Lautstärke, Lichtverhältnisse, etc.). Ein virtueller Sensor s definiert sich also wie folgt:

$$s = (p, r, k, t_0, t_1, T)$$

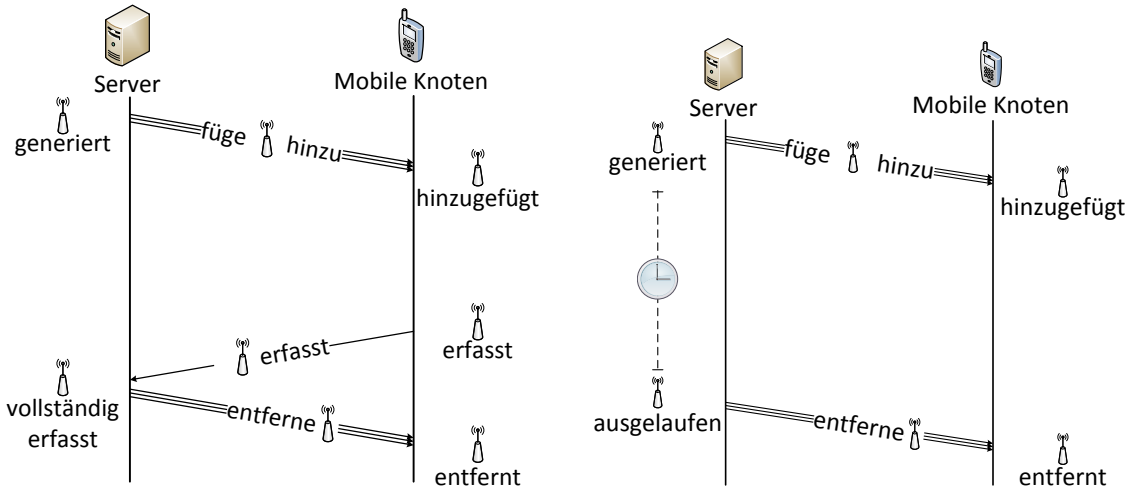
Mithilfe des Konzepts des virtuellen Sensors kann nun eine Aufgabe T definiert werden: Diese besteht aus einer Menge S von virtuellen Sensoren wobei deren Parameter jeweils frei belegt werden können:

$$T = (S),\ S = \{s_1, s_2, \ldots, s_n\}$$

Nachdem ein virtueller Sensor $s_i \in S$ von mindestens $s_i.k$ mobilen Knoten erfasst wurde, werden alle mobilen Knoten informiert, dass s_i erfasst wurde und keine weiteren Sensordaten für s_i gesammelt werden müssen, die Abfolge der dafür verschickten Nachrichten ist in Grafik 4.2a abgebildet.
Die aufgezeichneten Sensordaten werden dann gespeichert und für die spätere Auswertung durch den Benutzer gekennzeichnet.

Falls ein virtueller Sensor $s_j \in S$ nicht vor Ende seiner Zeitmarke $s_j.t_1$ von mindestens $s_j.k$ mobilen Knoten erfasst wurde, wird wiederum eine Nachricht an alle mobile Knoten geschickt, dass s_j aus dem System entfernt wird, allerdings werden die bereits für s_j gesammelten Daten nicht an den Benutzer weitergegeben, s_j wird viel mehr als „verpasst“ gekennzeichnet, die Abfolge der dafür verschickten Nachrichten ist in Grafik 4.2b abgebildet

Zusätzlich zur Ausführung und Verteilung von Aufgaben läuft im naiven System auch ein weiterer Prozess ab, die Positionserfassung der mobilen Knoten. Diese müssen zur Erfassung eines virtuellen Sensors ihre aktuelle Position kennen und können diese zudem dem zentralen Server mitteilen, wobei dies im Fall des naiven Systems keine Notwendigkeit darstellt, aber für die verbesserten Ansätze von Bedeutung sein wird.
Der Prozess des Aktualisierens der Position eines mobilen Knotens auf dem Server kann zudem durch ein Lokations-Update-Protokoll gesteuert werden.



(a) Nachrichtenabfolge im Fall der Erfassung eines virtuellen Sensors

(b) Nachrichtenabfolge im Fall des Auslaufens eines virtuellen Sensors, die Uhr symbolisiert den Ablauf eines Timeouts

Abbildung 4.2: Nachrichtenfolge der Erfassung eines virtuellen Sensors und Auslaufens eines virtuellen Sensors, dieser wird durch gekennzeichnet. Pfeile in dreifacher Ausführung symbolisieren den Versand der jeweiligen Nachricht an alle mobilen Knoten

4.1.1 Lokations-Update-Protokoll

Das Konzept des Update-Protokolls wurde aus [LR01] übernommen, wobei nur auf dem mobilen Knoten angesiedelte Protokolle, sogenannte Reporting-Protokolle, mit in das entwickelte System aufgenommen wurden, da ein Abfragen der Positionen der mobilen Knoten durch den Server, auch Querying genannt, im vorliegenden System nicht sinnvoll verwendbar wäre. Mit der Verwendung eines Update-Protokolls soll erreicht werden, dass eine geringere Anzahl an Positionserfassungen an den Server versendet wird, ohne die Effektivität des Systems zu verletzen, Ziel ist dabei die Minimierung des Energieaufwands für das Kommunikationssystem.

Die folgenden Update-Protokolle wurden dabei aus [LR01] übernommen:

- Simple: Bei jeder Positionserfassung wird eine Nachricht mit der aktuellen Position des mobilen Knotens an den zentralen Server geschickt.
- Time-based: Nach Ablauf eines Intervalls $t_{intervall}$ wird die aktuelle Positionsinformation des mobilen Knotens zum Server geschickt, damit ist eine periodische Aktualisierungsrate vorgegeben.
- Distance-based: Positionsinformationen werden erst übertragen, wenn ein Schwellwert d_{min} zwischen der letzten übertragenen Positionsinformation und der aktuellen Position überschritten ist.

Es ist überdies auch möglich, eine Kombination des Time-based und Distance-based Protokolls einzusetzen, wobei beide Protokolle auf ihre Zustimmung zu einer Übertragung der aktuellen Positionsinformation getestet werden und nur bei einer positiven Rückmeldung beider Protokolle Informationen an den Server geschickt werden.

4.1.2 Kommunikation

Eine Grundvoraussetzung für die Zwecke des Public Sensings ist die Fähigkeit der mobilen Knoten, erfasste Sensor- und Positionsdaten zum zentralen Server zu senden und Auftragsinformationen des zentralen Servers zu erhalten. Die Kommunikation zwischen Server und mobilen Knoten ist ein zentraler Bestandteil des grundlegenden Systems wie auch der im weiteren Verlauf verbesserten Systeme und wird daher gesondert eingeführt. Über den Zweck der Nachrichtenkommunikation hinaus besitzt das simulierte Kommunikationssystem zusätzlich die Möglichkeit, die für die Kommunikation aufgewandte Energie, wie in Kapitel 2.1.1 beschrieben, zu berechnen, womit für die spätere Evaluation die Metrik des Energieaufwands ermöglicht wird. In diesem Abschnitt sollen nun die verschiedenen Arten der Nachrichten im System vorgestellt werden:

Im vorliegenden System gibt es dabei mehrere verschiedene Arten von Nachrichten, die jeweils eigene Metadaten transportieren können, wobei der Absender immer bekannt ist und somit nicht zusätzlich in den Metadaten mit aufgeführt werden muss.
Die Tabelle 4.1 gibt dazu einen Überblick über die Nachrichten, wobei diese hinsichtlich der Senderichtung, also zum einen von den mobilen Knoten zum Server und zum anderen vom Server zu den mobilen Knoten, unterteilt sind.
Zur Registrierung der mobilen Knoten ist der Inhalt des initialen Datensatzes von Bedeutung: Dieser enthält Daten über die Fähigkeiten des jeweiligen mobilen Knotens, welche für die im weiteren Verlauf beschriebenen Ansätze zur Optimierung des Systems genutzt werden können.

Richtung	Nachrichtentyp	Metadaten
$C \rightarrow S$	Registrierung	initialer Datensatz
	GPS-Fixes	Positionsdaten
	virtueller Sensor erfasst	ID des virtuellen Sensors
$S \rightarrow C$	virtueller Sensor hinzugefügt	kompletter Datensatz des virtuellen Sensors
	virtueller Sensor gelöscht	ID des virtuellen Sensors

Tabelle 4.1: Nachrichtentypen im Public Sensing-System
Richtung deutet Absender und Empfänger einer Nachricht an:
$C \rightarrow S$ bedeutet: Nachricht wird von einem mobilen Knoten zum Server geschickt
$S \rightarrow C$ bedeutet: Nachricht wird vom Server zu einem mobilen Knoten geschickt

4.1.3 Erfassung virtueller Sensoren

Die Erfassung der virtuellen Sensoren durch mobile Knoten ist eine zentrale Aufgabe des Systems. Im Zuge dieser Arbeit erwarten dabei alle virtuellen Sensoren den gleichen Sensortyp, zudem haben die Sensorwerte der virtuellen Sensoren keine weitere Bedeutung, da in dieser Arbeit die Verteilung der Aufgaben auf die mobilen Knoten im Vordergrund stand. Von großer Bedeutung war aber die Erfassung der virtuellen Sensoren durch mobile Knoten an sich, deren Konzeption soll im Folgenden beschrieben werden.

Die Erfassung der virtuellen Sensoren gliedert sich dabei in zwei Fälle: Zum einen kann ein mobiler Knoten an der Position p_{mobil} einen virtuellen Sensor mit Position p Erfassungsradius r erfassen wenn gilt: $\|p - p_{mobil}\|_2 \leq r$, der mobile Knoten sich also im Erfassungsbereich des virtuellen Sensors befindet.
Dabei ist anzumerken, dass hierbei für die Position der mobilen Knoten immer die Position des Positionserfassungssystems (beispielsweise GPS) herangezogen wird und keine weitere Miteinbeziehung der möglichen Ungenauigkeit ϵ der GPS-Position erfolgt.

Zum anderen kann aber auch folgender Fall, der in Grafik 4.3 visualisiert ist, zu einer Erfassung eines virtuellen Sensors durch einen mobilen Knoten führen: Hierbei bewegt sich ein mobiler Knoten zwischen zwei GPS-Positionserfassungen von Position p_a zur Position p_b. Weder p_a noch p_b befinden sich innerhalb der Erfassungszone des virtuellen Sensors v und ein Vorgehen wie oben beschrieben ist nicht möglich. Daher wird nun ein Verfahren vorgestellt, mit dem auch die Erfassung in solche einem Kontext möglich ist, unter der Annahme, dass der mobile Knoten sich auf einer Linie bewegt. Ist dies nicht der Fall, so müssen andere Verfahren zum Einsatz kommen, bzw. möglicherweise auf die Erfassung in solch einem Fall verzichtet werden oder es muss ein kleineres Zeitintervall zwischen zwei Positionserfassung gewählt werden.

Eine mögliche Durchquerung eines Erfassungsbereiches wird nun dadurch entdeckt, dass der minimale Abstand d_{min} der Linie von p_a nach p_b zur Position p des virtuellen Sensors berechnet wird und verglichen wird, ob $d_{min} \leq r$ und damit eine Erfassung festgestellt wird.

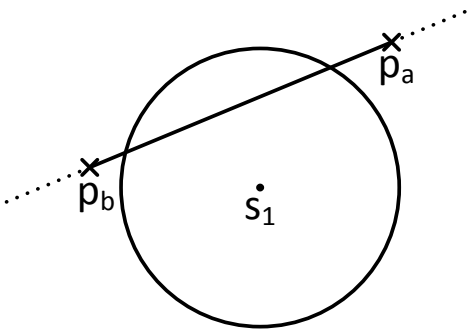


Abbildung 4.3: Erfassung eines virtuellen Sensors durch Analyse der Geraden durch p_a und p_b

Im Detail wird dabei eine Drehung des Koordinatensystem vorgenommen, sodass die Linie durch p_a nach p_b parallel zur x-Achse des Koordinatensystems verläuft. Dies geschieht unter Zuhilfenahme der Rotationsmatrix R_α aus Formel (4.1) welche einen Punkt $c = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix}$ in Vektorschreibweise mittels einer Matrixvektormultiplikation in den Punkt $c_{rotated}$ überführt wie Formel (4.2) darlegt.

$$(4.1)\quad R_\alpha = \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{pmatrix}$$

$$(4.2)\quad c_{rotated} = R_\alpha * c = \begin{pmatrix} c_1 \cdot \cos(\alpha) - c_2 \cdot \sin(\alpha) \\ c_1 \cdot \sin(\alpha) + c_2 \cdot \cos(\alpha) \end{pmatrix}$$

Der Winkel α lässt sich dabei im Zusammenhang mit der Linie durch p_a und p_b wie folgt berechnen, Grafik 4.4 zeigt dabei die einzelnen Komponenten der Rechnung auf. Hierbei wird wiederum die Annahme sichtbar, dass sich der mobile Knoten auf einer geraden Linie zwischen den Positionserfassungen bewegen muss.

$$\cos(\alpha) = \frac{|c|}{|b|}$$
$$\alpha = \text{acos}(\frac{|c|}{|b|})$$

Mithilfe von α und Formel (4.2) werden dann p_a und p_b zu $p_{a_{rotated}}$ und $p_{b_{rotated}}$ rotiert und nun kann leicht anhand einer Subtraktion der y-Koordinaten festgestellt werden, ob eine Erfassung vorlag oder nicht:

$$|p_{a_{rotated}}.y - p_{b_{rotated}}.y| \leq r$$

Mithilfe dieser Vorgehensweise können nun also überstrichene Erfassungszonen virtueller Sensoren detektiert werden, womit mittels einer diskreten Positionsbestimmung mit der bereits angedeuteten Einschränkung, dass sich der mobile Knoten auf einer geraden Linie zwischen zwei Positionserfassungen bewegen muss, die Effektivität einer kontinuierlichen Positionsbestimmung erreicht werden kann.

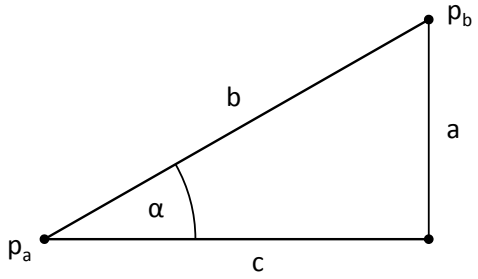


Abbildung 4.4: Visualisierung der Komponenten für die Berechnung des Winkels α

4.2 Adaptive Aufgabenverteilung

Der folgende Abschnitt beschäftigt sich mit Ansätzen zur Verbesserung der Effizienz der Aufgabenverteilung: Der bereits vorgestellte naive Ansatz schickt jeden hinzugefügten virtuellen Sensor an alle teilnehmende mobile Knoten, daraus ergibt sich die Anzahl der Nachrichten, die für das Hinzufügen aller virtueller Sensoren bei allen mobilen Knoten verschickt werden müssen als:

$$|\text{mobile Knoten}| * |\text{virtuelle Sensoren}|$$

Für das Entfernen der virtuellen Sensoren nach deren Erfassung oder Auslaufen wird dieselbe Anzahl an Nachrichten erneut verschickt. Daher soll die Anzahl der verschickten Nachrichten mit den nun vorzustellenden Ansätzen verbessert werden.

Die Grundidee, die zur Reduktion der Anzahl der verschickten virtuellen Sensoren führen soll ist, dabei die Folgende: Meldet ein mobiler Knoten zum Zeitpunkt t seiner aktuelle Position p_{node}, so kann, solange die Geschwindigkeit des mobilen Knotens nicht größer als eine dem Server bekannte Maximalgeschwindigkeit v_{max} ist, leicht berechnet werden, bis wohin der mobile Knoten sich bis zum Zeitpunkt t' bewegt haben kann. Die vom mobilen Knoten maximal zurückgelegte Wegstrecke s_{max} zum Zeitpunkt t' berechnet sich dabei als $s_{max}(t') = v_{max} * (t' - t)$.
Die mögliche Bewegung von der letzten bekannten Position des mobilen Knoten wird im Folgenden als Positionsunsicherheit bezeichnet, Grafik 4.5 zeigt dazu die Positionsunsicherheit des mobilen Knotens m zu drei verschiedenen Zeitpunkten t_1, t_2, t_3 wobei $t_1 < t_2 < t_3$ gilt. Dabei stellen die durch die Radien $s_{max}(t_1), s_{max}(t_2), s_{max}(t_3)$ definierten Kreisflächen jeweils die Menge der möglichen Positionen des mobilen Knotens m dar.

Der Startwert für s_{max} liegt bei der Positionsungenauigkeit ϵ des verwendeten Positionserfassungsverfahrens, da sich zum Zeitpunkt t der mobile Knoten innerhalb des Kreises der

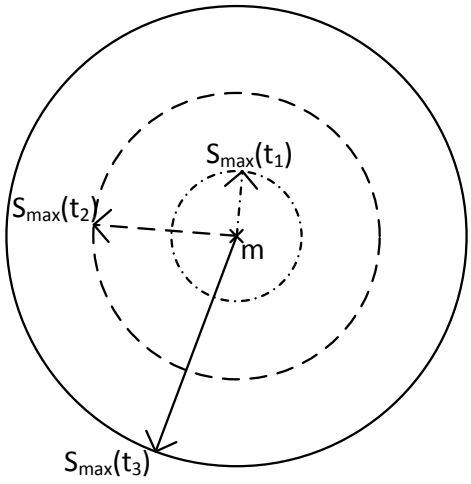


Abbildung 4.5: Visualisierung der Vergrößerung der Positionsunsicherheit zu drei verschiedenen Zeitpunkten $t_1 < t_2 < t_3$

Positionsungenauigkeit mit Radius ϵ (siehe auch Kapitel 2.2) befinden kann.
Mithilfe dieses Verfahrens sollen nun virtuelle Sensoren nur an diejenigen mobilen Knoten versendet werden, die diese potentiell auch erfassen können, anders als im naiven Fall erhalten nun also nicht mehr alle teilnehmenden mobilen Knoten einen virtuellen Sensor direkt bei dessen Erstellung, sondern erst nachdem der jeweilige mobile Knoten den virtuellen Sensor potentiell erfassen kann, bei diesem Vorgehen kann ein mobiler Knoten die Daten eines virtuellen Sensors auch nie zugeschickt bekommen wenn der mobile Knoten sich beispielsweise stets weit abwärts des virtuellen Sensors befindet.

Die vorgestellte Vorgehensweise wird im Folgenden als adaptive Verteilung der virtuellen Sensoren bezeichnet, wobei diese sich in eine Variante mit Unterstützung des zugrundeliegenden Karten-Graphen und in eine Variante mit reiner euklidischer Distanzberechnung gliedert.
Das Verfahren ist adaptiv, da es virtuelle Sensoren nur an mobile Knoten in der Nähe des virtuellen Sensors verteilt und damit virtuelle Sensoren mit vielen mobilen Knoten in der Umgebung oft verteilt, virtuelle Sensoren mit wenigen mobilen Knoten in der Nähe nur selten.

Eine mögliche Verbesserung ergibt sich zudem, wenn die mobilen Knoten das Distance-based Update-Protokoll verwenden: Ist von einem mobilen Knoten bekannt, dass dieser nur bei einer Überschreitung einer Abweichung von d_{min} neue Positionsinformationen an den Server sendet, so kann dem entsprechenden mobilen Knoten eine geringere Positionsungenauigkeit, nämlich d_{min} zugewiesen werden, woraus eine geringere Anzahl an verteilten virtuellen Sensoren resultiert.
Diese Optimierung kann aber nur dann erfolgen, wenn der entsprechende mobile Knoten eine Obergrenze der Abweichung zwischen zwei Positionserfassungen garantieren kann, da ohne diese Eigenschaft der mobile Knoten sich weiter als d_{min} von der letzten, dem Server bekannten, Position fortbewegen kann, ohne dass eine Positionsaktualisierung an den Server geschickt wird. Wird diese Eigenschaft verletzt, wäre die Folge dabei, dass möglicherweise virtuelle Sensoren nicht an den mobilen Knoten geschickt werden und damit die Effektivität des Gesamtsystems nicht mehr garantiert werden kann.

4.2.1 Euklidische Distanzen

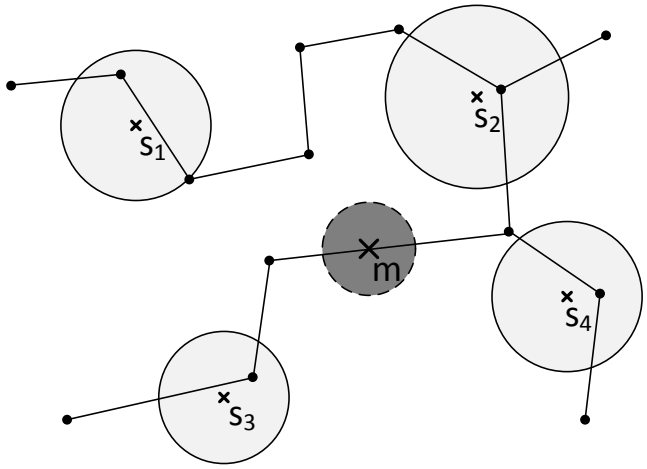
Ein erster Ansatz zur Reduktion der Nachrichtenlast setzt dabei kein Wissen über möglichen Wege der mobilen Knoten auf dem betrachteten Gebiet voraus, mit den vorliegenden Informationen wird also eine Kreisfläche beschrieben, in der sich der mobile Knoten befindet, ohne dass dem Server die genaue Position des Knotens bekannt ist, der Mittelpunkt des Kreises ist dabei p_{node} mit Radius s_{max}, wobei p_{node} der letzten, dem Server bekannten, erfassten Position des mobilen Knotens entspricht. Die maximale Positionsunsicherheit s_{max} wird also als direkte Wegstrecke zum Rand des Kreises mit der Geschwindigkeit v_{max} angenommen.
Bei jeder, dem Server mitgeteilten, Positionserfassung eines mobilen Knotens wird s_{max} auf den Startwert ϵ des verwendeten Positionserfassungssystems zurückgesetzt. Der Server berechnet fortlaufend für jeden mobilen Knoten dessen potentielle Distanz von der letzten

bekannten Position aus, wird dabei nun festgestellt, dass ein virtueller Sensor an der Position p_{sensor} mit der Reichweite $range$ einen kleineren Abstand zu p_{node} hat als s_{max}, also gilt: $\|p_{node} - p_{sensor}\|_2 - range \leq s_{max}$ so wird der betreffende virtuelle Sensor an den mobilen Knoten gesendet.
Zudem wird auf dem Server eine Abbildung von dem betreffenden virtuellen Sensor zu dem mobilen Knoten gespeichert, um im weiteren Ablauf auch das Entfernen der virtuellen Sensoren effizienter gestalten zu können:
Wird ein virtueller Sensor s_i auf dem Server entfernt, so erhalten nur diejenigen mobilen Knoten eine entsprechende gelöscht-Nachricht, die s_i auch erhalten hatten. Somit wird die Nachrichtenanzahl in beiden Phasen der Verteilung der virtuellen Sensoren verringert.

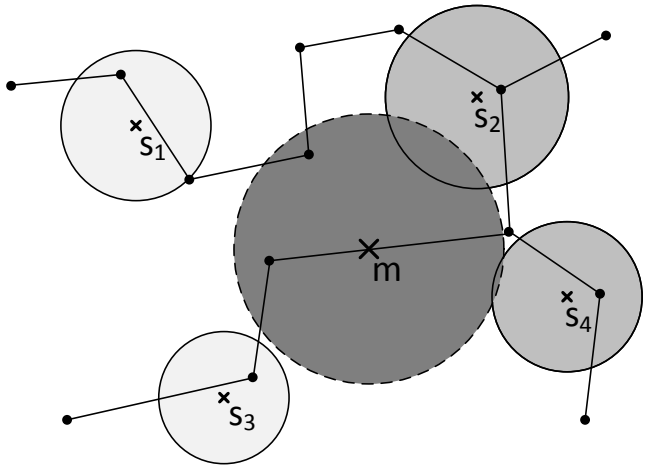
Der Vorteil der Strategie der adaptiven Verteilung mit euklidischen Distanzen liegt nun im Vergleich zum grundlegenden System darin, dass nicht mehr alle mobile Knoten alle virtuellen Sensoren zugeschickt bekommen, die Grafik 4.6 visualisiert dies. Dabei symbolisiert die dunkelgraue Kreisfläche um den mobilen Knoten m die potentiellen Positionen des mobilen Knotens.
In Grafik 4.6a ist die Ausgangssituation zum Zeitpunkt t_1 dargestellt, der Kreis der möglichen Position um die von m erfasste Position hat den Radius der Ungenauigkeit des verwendeten Positionserfassungsverfahrens.
Grafik 4.6b zeigt die Situation zum Zeitpunkt $t_2 > t_1$, hier wurden, durch die im Vergleich zum Zeitpunkt t_1 gestiegene Positionsunsicherheit, bereits die virtuellen Sensoren s_2 und s_4 an den mobilen Knoten geschickt, da dieser sich nun potentiell innerhalb der Erfassungszone dieser virtuellen Sensoren befinden kann.
In Grafik 4.6c wurden zum Zeitpunkt $t_3 > t_2$ auch die virtuellen Sensoren s_1 und s_3 an m verschickt, da die Positionsunsicherheit mittlerweile eine Entfernung zur letzten von m bekannten Position erreicht hat, die den Abstand von m zu s_1 überschreitet.
Mit Hilfe dieses Beispiels ist deutlich der Vorteil der adaptiven Verteilung zu erkennen: Wäre bei dieser Situation das grundlegende System aktiv, so würden alle vier virtuellen Sensoren sofort an m versendet, bei Anwendung der adaptiven Verteilung mit euklidischen Distanzen werden die virtuellen Sensoren erst dann verteilt, wenn der betreffende mobile Knoten sich potentiell in der Erfassungszone des jeweiligen virtuellen Sensors befinden kann.

4.2.2 Kartenbasierte Verteilung

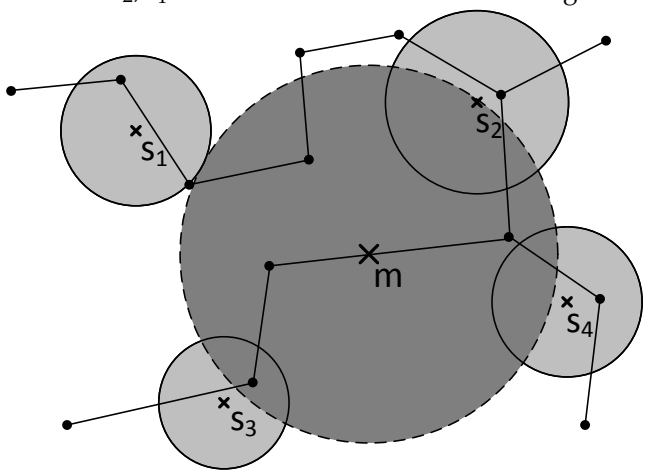
Im Gegensatz zur eben vorgestellten, einfachen Berechnung der euklidischen Distanzen zwischen den Positionen der mobilen Knoten und den virtuellen Sensoren soll nun auch das Wissen über die möglichen Wege der mobilen Knoten mit in die Berechnung der Distanz einbezogen werden. Dies schränkt die Verwendung dieses Ansatzes insofern ein, als dass der Graph der potentiellen Wege der mobilen Knoten bekannt sein muss und sich die mobilen Knoten nur geringfügig von dem vorgegebenen Kanten entfernen dürfen. Im Gegensatz zum gerade besprochenen Ansatz werden aber für jeden mobilen Knoten nun die Längen der Pfade zu allen aktiven virtuellen Sensoren berechnet und nicht mehr der euklidische Abstand verwendet.
Dies entspricht einer substantiellen Verbesserung der Effizienz bei der Anzahl übertragenen



(a) Initiale Situation direkt nach Positionserfassung des mobilen Knotens m zum Zeitpunkt t_1, m befindet sich innerhalb der dunkelgrauen Kreisfläche



(b) Zeitpunkt t_2: Der mobile Knoten befindet sich innerhalb der nun größeren dunkelgrauen Kreisfläche, die virtuellen Sensoren s_2, s_4 wurden dem mobilen Knoten zugestellt



(c) Zeitpunkt t_3: Dem mobilen Knoten wurden auch die virtuellen Sensoren s_1, s_3 zugestellt

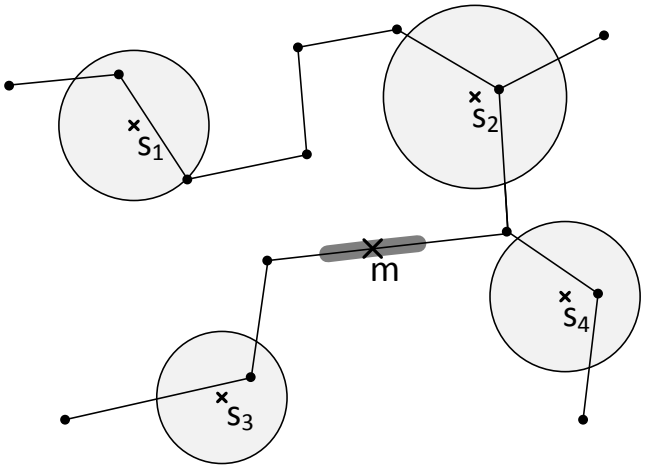
Abbildung 4.6: Visualisierung einer adaptiven Verteilung der virtuellen Sensoren mit euklidischen Distanzen, die hellgrauen Kreisflächen stellen jeweils die Erfassungszonen der noch nicht verteilten virtuellen Sensoren dar, dunkelgraue Erfassungszonen symbolisieren bereits an den mobilen Knoten verteilte virtuelle Sensoren

virtuellen Sensoren wie Kapitel 6 zeigt, zudem wird mit diesem Vorgehen eine bessere Approximation des Verhaltens der mobilen Knoten erreicht, da für diese nun der kürzeste Pfad als Grundlage der Berechnung, und damit als Worst-Case-Szenario, verwendet wird und nicht mehr, wie im Fall der euklidischen Distanzen, eine direkte Linie zum Rand der Kreisfläche der möglichen Positionen als Pfad des mobilen Knotens angenommene wird.

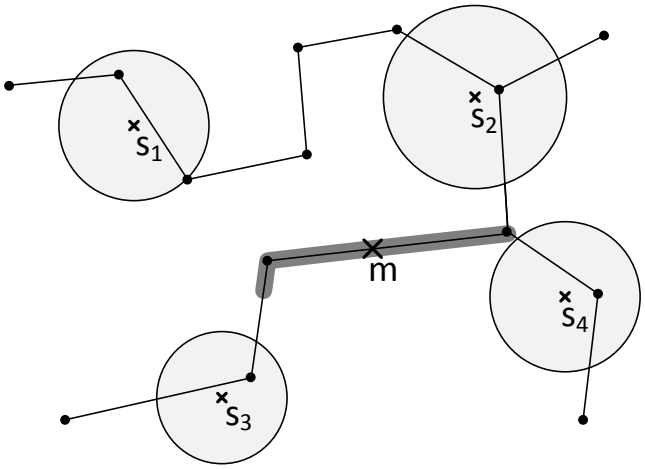
Ähnlich zur adaptiven Verteilung mit euklidischen Distanzen soll auch für die kartenbasierte Verteilung ein Beispiel dargelegt werden, welches den Vorteil dieses Ansatzes heraushebt: Grafik 4.7 zeigt dazu die verschiedenen Ausgangssituationen, der graue Bereich an den jeweiligen Graph-Kanten symbolisiert den Bereich, innerhalb dessen sich der mobile Knoten m befinden kann. Im Vergleich zu den euklidischen Distanzen ist der Unterschied hinsichtlich der Positionsunsicherheit direkt sichtbar: Für die euklidischen Distanzen ist die Positionsunsicherheit eine Kreisfläche mit der maximalen Bewegung als Radius. Im Fall des kartenbasierten Ansatzes ist die Positionsunsicherheit an die Kanten des zugrundeliegenden Graphen gebunden, hier entspricht die maximale Bewegung des mobilen Knotens der Wegstrecke von der initialen Position von m bis zu den jeweiligen Rändern des Bereichs der Positionsunsicherheit an den Kanten des Graphen.
Zur Vergleichbarkeit mit den euklidischen Distanzen werden die gleichen Zeitschritte und damit die gleiche maximale Bewegung verwendet, wie für das Beispiel der euklidischen Distanzen:
Grafik 4.7a zeigt dazu die Ausgangssituation mit der initialen Ungenauigkeit des Positionserfassungsverfahrens, noch wurden keine virtuellen Sensoren an m verteilt.
In Grafik 4.7b wird die Situation zum Zeitpunkt $t_2 > t_1$ dargestellt, weiterhin wurde kein virtueller Sensor an m verteilt, da sich dessen Positionsunsicherheit noch nicht innerhalb einer Erfassungszone eines virtuellen Sensors befindet. Dabei ist wiederum der potentielle Vorteil der kartenbasierten Verteilung im Vergleich zur Verteilung mit euklidischen Distanzen zu erkennen: Zum gleichen Zeitpunkt t_2 wurden m im Beispiel der euklidischen Distanzen bereits die virtuellen Sensoren s_2 und s_4 zugeschickt, bei Verwendung des kartenbasierten Ansatzes wurde noch kein virtueller Sensor verschickt.
Grafik 4.7c zeigt die Situation zum Zeitpunkt $t_3 > t_2$, hierbei ist die Positionsunsicherheit so groß geworden, dass nun m die virtuellen Sensoren s_2, s_3 und s_4 zugestellt wurden, dies stellt aber im Vergleich zur Verteilung mit euklidischen Distanzen immer noch eine Verringerung der Anzahl der verteilten virtuellen Sensoren dar, da bei der Verteilung mit euklidischen Distanzen zum Zeitpunkt t_3 bereits alle vier virtuellen Sensoren an m verschickt wurden.

Im Vergleich zur adaptiven Verteilung mit euklidischen Distanzen zeigt der kartenbasierte Ansatz also eine wesentliche Verringerung der Anzahl an verteilten virtuellen Sensoren, im Vergleich zum grundlegenden System ist die Verringerung der Anzahl verteilter virtueller Sensoren noch weitreichender, vor allem wenn in Betracht gezogen wird, dass in dem vorgestellten Beispiel erst nach Zeitpunkt t_2 überhaupt virtuelle Sensoren verteilt wurden, würden die anderen virtuellen Sensoren vor t_2 entfernt, müsste das grundlegende System jeweils die virtuellen Sensoren von m entfernen, die kartenbasierte Verteilung würde überhaupt keine weiteren Nachrichten schicken.

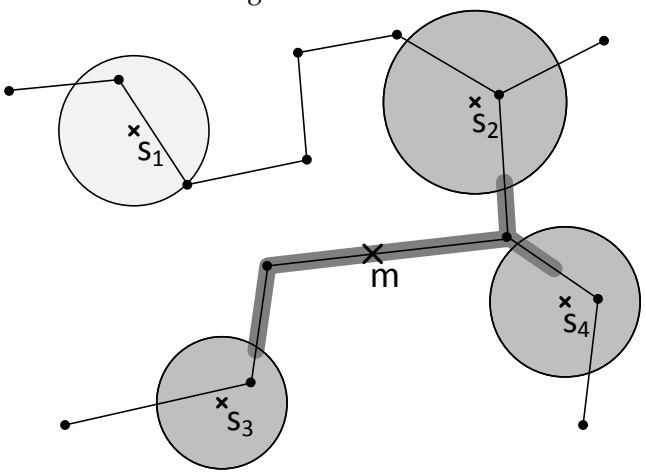
Zur Verwendung der kartenbasierten Verteilung der virtuellen Sensoren müssen dabei noch mehrere Teilprobleme gelöst werden, die in dem folgenden Abschnitt diskutiert werden.



(a) Initiale Situation direkt nach Positionserfassung des mobilen Knotens m zum Zeitpunkt t_1, m befindet sich innerhalb des dunkelgrauen Bereichs



(b) Zeitpunkt t_2: Der mobile Knoten befindet sich innerhalb des nun größeren dunkelgrauen Bereichs, es wurde noch kein virtueller Sensor zugestellt



(c) Zeitpunkt t_3: Dem mobilen Knoten werden die virtuellen Sensoren s_2, s_3, s_3 zugestellt

Abbildung 4.7: Visualisierung einer kartenbasierten adaptiven Verteilung der virtuellen Sensoren, die hellgrauen Kreisflächen stellen jeweils die Erfassungszonen der noch nicht verteilten virtuellen Sensoren dar, dunkelgraue Erfassungszonen symbolisieren bereits an den mobilen Knoten verteilte virtuelle Sensoren

Besonderheiten der kartenbasierten adaptiven Verteilung

Einen ersten Ansatz zur adaptiven Verteilung der virtuellen Sensoren auf die mobilen Knoten unter Zuhilfenahme des Graphen auf dem sich die mobilen Knoten bewegen liefert ein kürzeste Wege-Algorithmus mit einem Ausgangsknoten (single-source shortest-path), in der vorliegenden Arbeit wird dabei *Dijkstras Algorithmus* [Dij59] verwendet, mithilfe dessen bei Hinzufügen eines neuen virtuellen Sensors auf dem Server die Distanzen zu allen mobilen Knoten berechnet werden. Mithilfe dieser Information kann berechnet werden, wann mobile Knoten potentiell in die Erfassungszone eines virtuellen Sensors eintreten. Ausgestattet mit dieser Information kann der entsprechende virtuelle Sensor rechtzeitig an den mobilen Knoten übertragen werden. Voraussetzung für diesen Ansatz ist dabei, dass mobile Knoten den zuletzt von ihnen besuchten Graph-Knoten kennen und an den Server weiterleiten können und dass die virtuellen Sensoren auf Grundlagen des Graphen für die mobilen Knoten erstellt werden. Dies ist mit der Ausführung der Graph-Algorithmen zu begründen, da diese einen Start-Knoten als Eingabe erwarten.

Die Evaluation des vorgestellten Ansatzes enthüllt allerdings mehrere Schwächen, welche die Funktionalität und Effektivität des Ansatzes stören: Im vorliegenden Szenario ist es nicht möglich, die Bewegungsrichtung eines mobilen Knotens vorherzusagen und somit den nächsten von ihm besuchten Graph-Knoten vorherzusagen. Dies führt bei der Ausführung des Kürzeste-Wege-Algorithmus zu Problemen, da nur die Distanz von aktueller Position zu letztem besuchtem Graph-Knoten mit in die Distanzberechnung einbezogen werden kann, dadurch kann es aber zu fehlerhaften Berechnungen kommen: Hierbei wird das Szenario aus Grafik 4.8 verwendet, der mobile Knoten m bewegt sich in Richtung des Graph-Knotens v_2 und hat zuletzt v_1 besucht. Wird nun der kürzeste Weg zu Knoten v_3 berechnet, so ist dieser um den Abstand $\|m.p - v_1\|_2$ zu groß, eine Korrektur dieses Fehlers lässt sich nicht ohne aufwändige Pfadnachverfolgung erreichen. Diese führt allerdings zu unerwünschten Nachteilen bezüglich sowohl der Laufzeit- als auch der Speicherkomplexität, da nun Pfadinformationen gespeichert werden müssen sowie eine Rückverfolgung des Pfades bei jeder Distanzberechnung erfolgen muss.

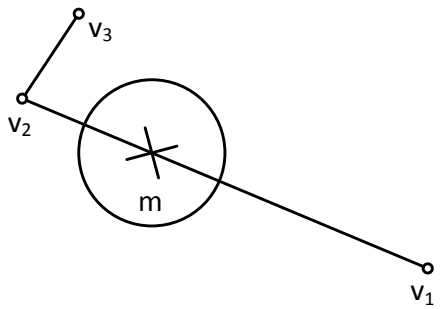


Abbildung 4.8: Beispielszenario für eine möglicherweise fehlerhafte Distanzberechnung mit mobilem Knoten m mit angedeuteter Ungenauigkeit des Positionserfassungsverfahrens, m bewegt sich auf v_2 zu

Ein weiteres, mit dem vorgestellten Ansatz nicht lösbares Problem zeigt sich, wenn die Situation in Grafik 4.9 betrachtet wird: s_1 ist dabei ein virtuellen Sensor mit Radius r_1, s_1 wurde auf Grundlage von v_1 erstellt, wobei v_1 der Zielkonten für *Dijkstras Algorithmus* ist, m ist ein mobiler Knoten mit einer Positionsungenauigkeit von r_2 mit zuletzt besuchtem Knoten v_6. Wird nun die Distanzberechnung von v_6 zu v_1 gestartet, wird das Ergebnis 13 betragen. Wendet man den beschriebenen Distanzausgleich für den mobilen Knoten an, subtrahiert also den Abstand von v_6 zu m (1) und addiert zusätzlich den Abstand von s_1 zu v_1 (ebenfalls 1) so erhält man als Abstand von m zu s_1 $13 + 1 - 1 = 13$.
Dies führt zu einer fehlerhaften Ausführung der Aufgabe, da sich m zwar in der Erfassungszone von s_1 befindet, diesen virtuellen Sensor aber vom Server noch nicht zugestellt bekommen hat, da sich m und s_1 laut der Distanzberechnung noch weit auseinander befinden. Da auch dieses Szenario abgedeckt werden soll, wird nun ein verbesserter Ansatz vorgestellt.

Der verbesserte Ansatz beruht dabei auf der Entscheidung für einen paarweise kürzeste Wege-Algorithmus (all-pairs shortest-path): Anstelle der einzelnen Graph-Knoten für mobile Knoten und virtuelle Sensoren tritt eine Menge von Graph-Knoten: Für jeden Schnittpunkt mit dem Kreis, welcher bei den mobilen Knoten durch deren Position und der Positionsungenauigkeit des simulierten GPS-Systems, bei den virtuellen Sensoren durch deren Position und den Radius der Erfassungszone definiert wird, und einer Kante des Graphen wird an der Position des Schnittpunktes ein neuer Graph-Knoten eingefügt. Dies ermöglicht zwar eine Vermeidung des Szenarios aus Graphik 4.9, Probleme können aber immer noch auftreten, wenn der Kreis um den virtuellen Sensor zwar keine Kante schneidet, aber einen minimalen Abstand zu einer Kante besitzt, der kleiner ist als die Positionsungenauigkeit der mobilen Knoten: Dieses Szenario ist in Grafik 4.10 dargestellt wobei hierbei ein Problem genau dann vorliegt, wenn der mobile Knoten sich durch die Positionsungenauigkeit innerhalb

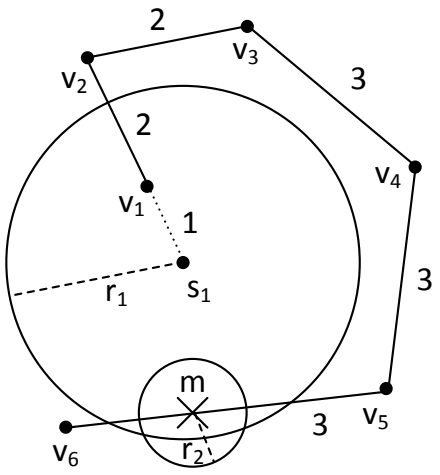


Abbildung 4.9: Beispielszenario für möglicherweise fehlerhafte Distanzberechnung mit berechnetem Abstand 13 obwohl m sich in der Erfassungszone von s_1 befindet

der Schnittfläche der Kreise um den virtuellen Sensor und um den mobilen Knoten selbst befindet.

Die Lösung ist ebenso schnell ersichtlich: Der Kreis um den virtuellen Sensor muss den Radius $r_3 = r_1 + r_2$ mit r_1 als Radius der Erfassungszone des virtuellen Sensors und $r_2 = \epsilon$ als Positionsungenauigkeit des mobilen Knoten besitzen, damit auch dieser potentielle Fehlerfall gelöst werden kann.

Damit kann nun eine neue Distanzberechnung gestartet werden, welche bei mobilen Knoten für jeden für diesen Knoten neu hinzugefügten Graph-Knoten eine Analyse aller virtueller Sensoren mit jeweils allen hinzugefügten Graph-Knoten beginnt und die Ergebnisse danach zusammenführt. Dabei werden nur die Distanzen berechnet, wobei für mobile Knoten, die sich im inneren einer Erfassungszone befinden, negative Distanzen ausgegeben werden, da diese keine Distanz mehr bis zum Erreichen der Erfassungszone zurücklegen müssen. Eine Berechnung der Pfade der mobilen Knoten muss dabei zur Distanzberechnung nicht vorgenommen werden, dies verringert den Berechnungsaufwand. Wie bereits angedeutet werden auch für die mobilen Knoten neue Graph-Knoten angelegt: Für jeden mobilen Knoten werden die Schnittpunkte mit Kanten und dem durch die Ungenauigkeit des Positionserfassungsverfahrens definierten Kreis berechnet und ebenfalls in neue Graph-Knoten konvertiert, diese sind dabei wahlweise Eingabe, beispielsweise wenn der kürzeste Weg von einem mobilen Knoten zu einem virtuellen Sensor gefunden werden soll, oder Ziel, wenn die Distanzberechnung von einem virtuellen Sensor zu dem mobilen Knoten gestartet wird.

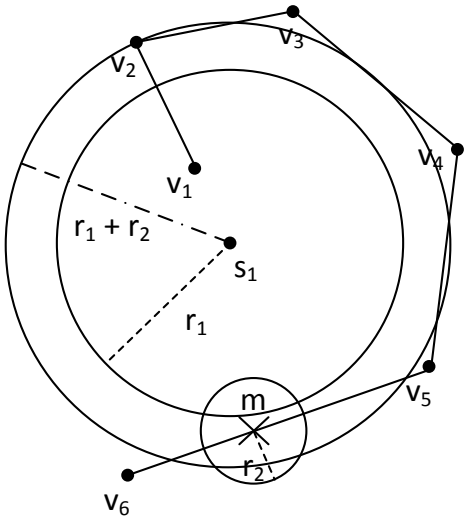


Abbildung 4.10: s_1 ist ein virtueller Sensor mit Radius r_1, der schlussendlich betrachtete Kreis um s_1 hat dabei den Radius $r_1 + r_2$ damit auch eine Positionserfassung von m in der Schnittfläche der Kreise um s_1 mit Radius r_1 und des Kreises um m mit Radius $r_2 = \epsilon$ eine Verteilung des virtuellen Sensors und damit eine Erfassung möglich macht.

Damit der beschriebene Ansatz auch effizient umgesetzt werden kann, müssen noch mehrere Detailprobleme gelöst werden, wie die effiziente Berechnung der kürzesten Distanzen: Für jeden Graph-Knoten müssen die Distanzen zu allen anderen Graph-Knoten bekannt sein, was zwar mit einem paarweise kürzeste Wege-Algorithmus leicht berechnet werden kann, allerdings nur mit einer relativ schlechten Komplexität von $\Theta(n^3)$ mit n als Anzahl der Graph-Knoten. Zudem muss diese Berechnung bei jedem Hinzufügen eines virtuellen Sensors und bei jeder Positionsaktualisierung eines mobilen Knoten durchgeführt werden, was bei größeren Graphen nur noch mit großen Laufzeiten möglich ist. Daher wird in Kapitel 5.3.2 zuerst der verwendete Algorithmus gefolgt von einem optimierten Ansatz für die Aktualisierung der Distanzen vorgestellt.

Des Weiteren müssen auch noch weitere Probleme gelöst werden, wie beispielsweise das Auffinden der benachbarten Kanten eines virtuellen Sensors, dies wird in Kapitel 5.3.2 detailliert dargelegt, worin zudem auf weitere Quellen von Komplexitätsproblemen bei der Verwaltung der Kanten und Berechnung der Schnittpunkte sowie der Lösung dieser Probleme eingegangen wird.

4.3 Adaptive Positionserfassung

Im bisher beschriebenen Ablauf führen mobile Knoten in regelmäßigen, kleinen (z.B. 10 s) Intervallen eine GPS-Positionserfassung aus. Dies ist aber nur dann notwendig, wenn überhaupt virtuelle Sensoren in der unmittelbaren Umgebung um die aktuelle Position des mobilen Knotens platziert wurden, ist dies nicht der Fall, so kann das Intervall der GPS-Positionserfassung auch größer sein.

Auf die Beobachtung aufbauend wurde eine Adaptive Positionserfassung entwickelt: Dabei wird das GPS nur aktiviert, wenn sich virtuelle Sensoren potentiell in der Nähe des mobilen Knoten befinden, die Aktualisierungsrate des GPS passt sich also adaptiv der Entfernung zu virtuellen Sensoren an. Eine adaptiven Positionserfassung läuft dabei wie folgt ab: Nach einer GPS-Positionserfassung wird der Abstand zu allen bekannten virtuellen Sensoren berechnet, abzüglich deren Erfassungsreichweite, sei dieser d_{min}. Mithilfe der maximalen Geschwindigkeit eines mobilen Knotens v_{max} wird damit die Zeit $t_{timeout} = \frac{d_{min}}{v_{max}}$ berechnet, ab der ein mobilen Knoten frühestens die Erfassungszone eines virtuellen Sensors betreten kann, der Ablauf ist in Pseudocode in Algorithmus 4.1 dargelegt. Dieser Vorgang wird wiederholt, sobald ein virtueller Sensor dem mobilen Knoten bekannt wird, bzw. wenn ein virtueller Sensor nach Erfassung oder Ablauf der Erfassungszeit (t_1) vom mobilen Knoten entfernt wird, um die Effektivität dieses Ansatzes zu gewährleisten.

Das Zeitintervall zwischen zwei adaptiven Positionserfassungen kann dabei natürlich nicht kleiner als das vorgegebene GPS-Update-Intervall sein, dies kann auftreten, wenn sich ein mobiler Knoten an eine Erfassungszone auf eine geringe Distanz annähert, in diesem Fall wird die vorgegebene Aktualisierungsrate des GPS als $t_{timeout}$ verwendet.

Die Verwendung der adaptiven Positionserfassung hat aber auch Einfluss auf die Funktionsweise der Lokations-Update-Protokolle aus Abschnitt 4.1.1:

Algorithmus 4.1 Ablauf einer adaptiven Positionserfassung

procedure ADAPTIVEGPSPOSITIONING
  gpsFix = getGPSFix()
  $d_{min} = \infty$
  for all virtualSensors **as** virtualSensor **do**
    $d_{current}$ = virtualSensor.p.distance(gpsFix.position) - virtualSensor.r
    if $d_{current} < d_{min}$ **then**
      $d_{min} = d_{current}$
    end if
  end for
  $t_{timeout} = \frac{d_{min}}{v_{max}}$
  scheduleGPS(max($t_{timeout}, t_{updateRate}$))
end procedure

Da die adaptive Positionserfassung keine feste Aktualisierungsrate garantiert, verändert sich die sowohl die Semantik des Time-based als auch des Distance-based-Update-Protokolls: Das Time-based-Protokoll garantiert für die herkömmliche Positionserfassung eine feste Aktualisierungsrate der Position des mobilen Knotens auf dem Server. Dies kann bei der adaptiven Positionserfassung nicht mehr gewährleistet werden, da die Aktualisierungsrate der Position variiert.
Aus demselben Grund verändert sich auch die Semantik des Distance-based-Protokolls, da keine Garantie mehr gegeben werden kann, dass der mobile Knoten eine Überschreitung des Abstands d_{min} im Vergleich zur auf dem Server bekannten Position erkennt. Damit fällt bei der Verwendung der adaptiven Positionserfassung auch die Optimierung der adaptiven Verteilung der virtuellen Sensoren, bei welcher die Unsicherheit der Position bei d_{min} abgeschnitten wird, weg, da sich der mobile Knoten auch weiter als d_{min} von der letzten bekannten Position entfernen kann.

Mithilfe dieses Ansatzes werden zwar nicht alle unnötigen GPS-Positionserfassungen außerhalb der Erfassungszone eines virtuellen Sensors unterdrückt, da potentiell weiterhin Positionserfassungen abseits eines virtuellen Sensors durchgeführt werden, da der Ansatz keine Pfadvorhersage treffen kann, aber die Anzahl der Positionserfassungen und damit die dafür aufzubringende Energie kann substantiell verringert werden, wie in Kapitel 4 aufgezeigt wird. Zusätzlich zur alleine durch die reduzierte Anzahl an GPS-Positionserfassungen gesparten Energie wird darüber hinaus auch weniger Energie zum Versenden der Positionsdaten an den Server aufgebracht.

5 Implementierung

Das folgende Kapitel soll nun einen Überblick über die Implementierung der Simulation unter Zuhilfenahme des Netzwerksimulators *The ONE* sowie über die Entscheidungen bezüglich der Implementierung des Entwurfs geben.

5.1 Simulator

Die vorliegende Arbeit stützt sich im Bereich der Evaluation vor allem auf Simulationen, da eine reale Evaluation aufgrund der zu großen Anzahl an unterschiedlichen Szenarien nicht in Betracht gezogen werden konnte. Zu dem Zweck der Simulation wird der Netzwerksimulator *The ONE* (The Opportunistic Network Environment Simulator) [KOK09] verwendet. Dieser ist ursprünglich zur Simulation von Ad-Hoc Netzwerken und verzögerungstoleranten Netzwerken konzipiert und wurde im Zuge dieser Arbeit um Komponenten für die Simulation im Sinne des Public Sensings erweitert. Der Simulator erlaubt die Ausführung verschiedener Applikationen auf an der Simulation teilnehmenden Knoten, dabei wurden im Zuge dieser Arbeit zweierlei Applikation für zum einen mobile Knoten und zum anderen für den zentralen Server entwickelt.
The ONE basiert auf einem diskreten Simulationskonzept, pro simulierte Sekunde wird der neue Simulationsstand dementsprechend n mal neu berechnet, wobei n eine vor Simulationsbeginn bestimmte Anzahl ist.
Zur Implementierung eigener Applikationen stellt *The ONE* die Basisklasse `Application` zur Verfügung, diese definiert die Methode `update` welche pro Simulationszyklus einmal pro Knoten aufgerufen wird und damit dem Knoten die Möglichkeit gibt, eigene Aktionen, wie beispielsweise Kommunikationsprozesse, durchzuführen.

5.2 Grundlegendes System

Die Public Sensing-Simulation wurde als `Application` in den bestehenden Rahmen von *The ONE* eingefügt, dies geschieht mittels der Klasse `PublicSensingApplication`. Diese erstellt für jeden mobilen Knoten eine eigene `ClientApplication` sowie für die gesamte Simulation genau eine `ServerApplication`, die für alle mobilen Knoten zuständig ist, die gesamte Klassenhierarchie ist in Grafik 5.1 zu sehen.

Zudem reicht die `PublicSensingApplication`-Klasse Aufrufe der bereits beschriebenen `update`-Methode von *The ONE* an den Server bzw. an die mobilen Knoten weiter, damit diese eigenständig Operationen wie die Positionserfassung oder das Erfassen virtueller Sensoren durchführen können.
Die `ServerApplication` wird dabei nicht auf einem an der Simulation teilnehmenden Knoten ausgeführt, wie es das *The ONE*-Framework vorsieht, sondern als eigenständige Klasse ohne Knoten implementiert, da dies dem Konzept des Servers entspricht, der als zentrale Instanz und nicht als mobiler Knoten fungiert. Dies führte zur Entscheidung, nicht auf das in *The ONE* integrierte Kommunikationsprinzip zurückzugreifen, dass zudem auf Kommunikationskanälen beruht, die nur innerhalb einer bestimmten Reichweite funktionieren. Stattdessen wurde wiederum ein eigenständiges Kommunikationsprinzip geschaffen, mit dem Server und mobile Knoten miteinander kommunizieren können, dieses Prinzip beruht dabei auf einer Simulation einer 3G-Verbindung, die zudem mit einem Energiemodell, wie in Kapitel 2.1.1 beschrieben, versehen wurde. Für die in Kapitel 4.1.2 vorgestellten Nachrichtentypen wurde aufbauend auf der in *The ONE* integrierten Nachrichtenklasse `Message` eine erweiterte Klasse angelegt, diese besitzt zusätzlich zur existierenden Klasse die Möglichkeit, die verschiedenen Adressaten einer Nachricht anzugeben.
Darüber hinaus wurde eine Report-Möglichkeit zur Evaluation der durch die Simulation gesammelten Daten geschaffen, welche die Grundlage für die Evaluation legen. Dabei wurde auf die in *The ONE* integrierten Möglichkeiten zur Erstellung applikationsspezifischer Daten zurückgegriffen.

Zur Kalibrierung der Simulation können für alle in Kapitel 6.1 vorgestellten Parameter Werte vorgegeben werden, welche den Ablauf der Simulation beeinflussen.

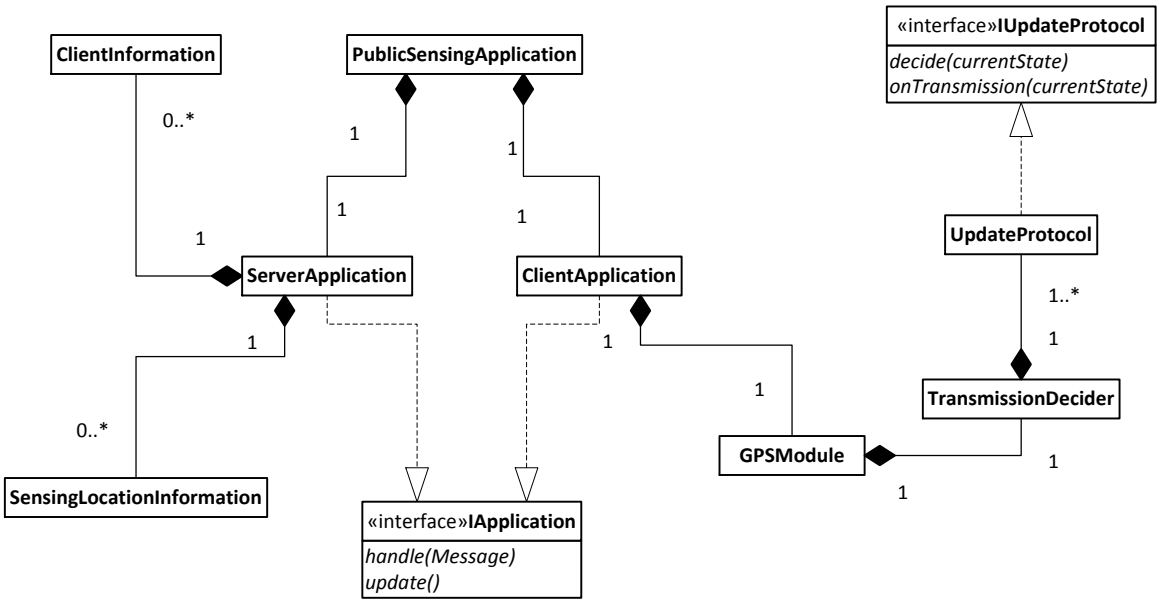


Abbildung 5.1: Klassenhierarchie der `PublicSensingApplication` inklusive aller relevanten Unterklassen für die `ServerApplication` und `ClientApplication`

5.2.1 Simulator-Implementierung

Das folgende Kapitel soll nun einen kurzen Überblick über die Implementierung der einzelnen Komponenten des Public Sensing-Systems, also die mobilen Knoten und der Server, geben.

Server

Zunächst soll auf die Details der Implementierung hinsichtlich der Verwaltung der virtuellen Sensoren und der mobilen Knoten eingegangen werden: Für jeden mobilen Knoten wird eine Instanz der Klasse `ClientInformation` angelegt, welche die für einen mobilen Knoten spezifischen Eigenschaften wie die aktivierten Update-Protokolle, die bereits an diesen Knoten versendeten virtuellen Sensoren sowie die aktuelle Position beinhaltet.
Ein ähnliches Vorgehen wird bei den virtuellen Sensoren angewandt: Die Klasse `SensingLocationInformation` kapselt neben einem virtuellen Sensor weitere Daten, wie die Anzahl der bereits für diesen Sensor eingetroffenen Messwerte.

Der Zusammenhang zwischen der `ServerApplication` und den Klassen `ClientInformation` und `SensingLocationInformation` ist auch in der Klassenhierarchie in Grafik 5.1 sichtbar.

Die Umsetzung der naiven Verteilung der virtuellen Sensoren lässt sich leicht bei der Generierung eines virtuellen Sensors realisieren: Der betreffende virtuelle Sensor wird an alle registrierten mobilen Knoten versendet und für jeden mobilen Knoten wird in der zugehörigen `ClientInformation` der Empfang des betreffenden virtuellen Sensors vermerkt. Wird ein virtueller Sensor von einem mobilen Knoten erfasst, so wird bei dem entsprechenden Eintrag des mobilen Knotens der gegebene virtuelle Sensor entfernt, auch wenn noch nicht k Erfassungen für den virtuellen Sensor erfolgten. Dies hat den Hintergrund, dass ein von einem mobilen Knoten erfasster virtueller Sensor nur genau einmal von diesem mobilen Knoten erfasst werden soll und deswegen direkt nach der Erfassung auf dem mobilen Knoten gelöscht wird, bei Entfernung des virtuellen Sensors auf dem Server muss also auch keine Nachricht an den erfassenden mobilen Knoten gesendet werden, welche das Entfernen des virtuellen Sensors symbolisiert.

Mobile Knoten

Wie in der Klassenhierarchie in Grafik 5.1 ersichtlich, wurden die mobilen Knoten als Kind-Elemente der PublicSensingApplication entworfen, die eigenständige Aktionen zu diskreten Zeitpunkten durchführen können. Dabei gliedern sich die Aufgaben des mobilen Knoten in zwei Teilbereiche: Positionsbestimmung und Erfassung virtueller Sensoren. Das Klassendiagramm in Grafik 5.1 zeigt dazu den Aufbau der `ClientApplication` inklusive des simulierten GPS und dessen Untermodule.

Positionserfassung Das simulierte GPS wurde in einem eigenen Untermodul des mobilen Knoten untergebracht, der Klasse `HostGPSModule`. Diese Klasse implementiert das in Kapitel 2.2 beschriebene Energiemodell und verwendet das beschriebene Berichtssystem von *The ONE* zur späteren Auswertung der für die Positionserfassung aufgebrachten Energiemenge. Die in Kapitel 2.2.1 beschriebene Ungenauigkeit der Positionserfassung wurde übernommen, wobei für ϵ der Wert 10 m angenommen wird. Zusätzlich zur Ungenauigkeit der Positionserfassung wurde auch die Verzögerung bei der Aktualisierung der GPS-Position mit berücksichtigt, dies wurde durch eine der Formel (2.4) entsprechende Verzögerung hinsichtlich der Simulationszeit erreicht.
Das simulierte GPS verwendet dabei entweder eine feste Update-Rate, wenn es im normalen Modus betrieben wird, oder eine variable Update-Rate im Modus der adaptiven Positionserfassung, diese wird nach Algorithmus 4.1 berechnet.

Lokations-Update-Protokoll Nicht zuletzt wurden die in Kapitel 4.1.1 beschriebenen Update-Protokolle implementiert, wobei diese aus dem eigentlichen GPS-Modul ausgelagert wurden, um eine spätere Erweiterung um weitere Protokolle leicht zu realisieren. Dazu wurde das Interface `IUpdateProtocol` entworfen, dass eine Entscheidung über die Versendung der aktuellen Positionserfassung trifft, wobei jedes Update-Protokoll dieses Interface implementiert. Das Interface definiert dabei die folgenden Methoden:

- `decide`: Diese Methode wird bei der Entscheidungsfindung mit dem aktuellen Zustand (Position, zuletzt besuchter Graph-Knoten, etc.) des mobilen Knotens aufgerufen und gibt einen Wahrheitswert zurück, der bei einer Zustimmung zur Übertragung positiv ist, bei Ablehnung der Übertragung dementsprechend negativ. Ist das aktuelle Update-Protokoll deaktiviert, so soll ein positiver Wahrheitswert zurückgegeben werden.
- `onTransmission`: Diese Methode wird bei der Zustimmung aller Update-Protokolle und damit bei der Übertragung der aktuellen Position aufgerufen, sie hat als Übergabeparameter wieder den aktuellen Zustand des mobilen Knotens. Diese Methode erlaubt es den Update-Protokollen ihre internen Variablen auf den Anfangszustand einer Übertragungsperiode zurückzusetzen.

Wird von allen Update-Protokollen eine positive Entscheidung getroffen (entweder ist dann das betreffende Update-Protokoll deaktiviert oder die aktuelle Situation erlaubt eine Übertragung) werden die jeweiligen Variablen der Update-Protokollen durch die in dem Interface `IUpdateprotocol` spezifizierte `onTransmission`-Methode zurückgesetzt, es wird also ein neuer Übertragungszyklus begonnen.

Erfassung virtueller Sensoren Die Grundlagen der Erfassung der virtuellen Sensoren wurden in Kapitel 4.1.3 beschrieben und in die `ClientApplication` implementiert. Diese Erfassungsstrategie wird nun nach jeder GPS-Positionserfassung angewandt, wobei diese wiederum zu den diskreten Zeitpunkten des Simulators erfolgen können. Nach der Erfassung eines virtuellen Sensors wird dieser vom mobilen Knoten gelöscht, damit keine Doppelerfassungen möglich sind, sowie eine Erfassungsnachricht an den Server versendet.

5.3 Adaptive Aufgabenverteilung

Das folgende Kapitel soll die Implementierungsentscheidungen der adaptiven Aufgabenverteilung vorstellen, wobei wiederum zuerst die Implementierung für den Simulator *The ONE* vorgestellt wird, bevor die zu lösenden Probleme genannt und deren Lösungen vorgestellt werden.

5.3.1 Simulator-Implementierung

Analog zur Vorstellung der Implementierung des grundlegenden, naiven Systems soll nun auf die Implementierung der adaptiven Aufgabenverteilung für den *The ONE*-Simulator eingegangen werden. Dabei wird im Folgenden nur auf die Implementierung der Server-Applikation eingegangen, da sich für die mobilen Knoten bei der Verwendung der adaptiven Verteilung keine Veränderungen ergeben.

Bei der Verwendung der adaptiven Verteilung werden in der Klasse `ClientInformation`, zusätzlich zu den bereits beim grundlegenden System gespeicherten Attribute eines mobilen Knoten, die nächstgelegenen virtuellen Sensoren in einem Heap gehalten, um einen schnellen Zugriff auf die naheliegenden virtuellen Sensor zu garantieren.
Bei Verwendung der kartenbasierten Verteilung werden in der Klasse `ClientInformation` zudem die durch diesen mobilen Knoten hinzugefügten Graph-Knoten gespeichert, um die Distanzberechnung zwischen virtuellen Sensoren und mobilen Knoten durchführen zu können und um diese Graph-Knoten nach einer erneuten Positionserfassung des mobilen Knotens wieder löschen zu können.

Dasselbe Vorgehen wird für die Klasse `SensingLocationInformation` angewandt, auch hier werden bei Verwendung der kartenbasierten adaptiven Verteilung die durch diesen virtuellen Sensor hinzugefügten Graph-Knoten zur Verwendung in der Distanzberechnung und zur Restrukturierung des Graphen nach Erfassung oder Auslaufen des virtuellen Sensors gespeichert.

Aufbauend auf die naive Verteilung wird die adaptive Verteilung mit Verwendung entweder der euklidischen Distanzen oder des kartenbasierten Ansatzes implementiert: Dabei wird nach dem Hinzufügen eines virtuellen Sensors auf dem Server für jeden mobilen Knoten dessen Distanz von der letzten, dem Server bekannten erfassten Position bis zum Beginn der Erfassungszone des virtuellen Sensors berechnet und in den zugehörigen Heap eingefügt. Zu jedem diskreten Zeitpunkt wird dann überprüft, ob der erste virtuelle Sensor im Heap potentiell in Reichweite ist. Ist dies der Fall, wird der virtuelle Sensor an den mobilen Knoten geschickt und aus dem Heap gelöscht. Nach einer vom Server empfangenen Positionserfassung des mobilen Knotens wird der Heap komplett neu aufgebaut, wobei alle bereits an den betreffenden Knoten gesendeten und von diesem Knoten erfassten virtuelle Sensoren nicht in den Heap aufgenommen werden.
Wird ein virtueller Sensor nach Erfassung von k mobilen Knoten oder dem Auslaufen der Erfassungszeit vom Server gelöscht, so muss die Nachricht, die dies signalisiert, nun auch nur an genau die Menge an mobilen Knoten gesendet werden, die diesen virtuellen Sensor

auch zugeschickt bekommen hatten und noch nicht erfasst haben.
Zusätzlich muss der Server für jeden Knoten dessen potentielle Bewegung, also die aktuelle Positionsunsicherheit, berechnen, dies wird zu jedem diskreten Zeitpunkt vorgenommen, wobei die mögliche Bewegung mittels der maximalen Geschwindigkeit des mobilen Knotens berechnet wird, wie in Kapitel 4.2 beschrieben.

Für die kartenbasierte adaptive Verteilung muss die Distanzberechnung für den Heap-Aufbau verändert werden, so dass nun die Distanzen über die Pfadlängen und nicht mehr über die euklidischen Distanzen zwischen mobilen Knoten und Erfassungszonen der virtuellen Sensoren berechnet werden. Dies erfordert teils weitreichende Veränderungen, da zwar *The ONE* ein Graph-Konzept aufbauend auf Adjazenzlisten besitzt, welches sich allerdings als ungenügend erwies, um die aufwendigen Graph-Transformationen, wie in Kapitel 4.2.2 beschrieben, durchführen zu können. Aus diesem Grund wird für die Implementierung des kartenbasierten Ansatzes auf die Graph-Bibliothek jGraphT [JGr12] zurückgegriffen, wobei der initiale Graph dem internen Graph von *The ONE* entnommen wird und alle Modifikationen alleine in dem dann eigenständigen jGraphT-Graphen vorgenommen werden. Damit wird auch die Implementierung des Algorithmus von Floyd-Warshall aus Algorithmus 5.1 erleichtert, da mit jGraphT auch der Zugriff auf einzelne Kanten des Graphen möglich ist, was bei einer Adjazenlisten-Darstellung nicht in konstanter Zeit möglich ist.
Des Weiteren wird für das Auffinden der Schnittpunkte des Kreises um virtuelle Sensoren oder mobile Knoten mit Graph-Kanten auf javaGeom [Jav12], eine speziell für diese Operationen optimierte Bibliothek, zurückgegriffen, wobei das Testen auf potentielle Schnittpunkte des Kreises mit Kanten ohne die Bibliothek durchgeführt wird, da diese zum Test Schnittpunkte berechnet, was eine potentiell teurere Aktion darstellt als der direkte Test auf Schnittpunkte ohne diese sofort zu berechnen, diese Berechnung wird zusätzlich implementiert um Komplexitätsproblemen vorzubeugen.

5.3.2 Entwurfsumsetzung

Der Entwurf der kartenbasierten adaptiven Verteilung beinhaltet einige Probleme, die bei einer direkten Implementierung zu einer nicht effizient durchführbaren Simulation führen, wie die Ausführung der Distanzberechnung oder die Auffindung der von der Erfassungszone eines virtuellen Sensors geschnittenen Graph-Kanten. Dieses Kapitel soll daher nun ausführen, wie diese Probleme im Zuge dieser Arbeit effizient gelöst werden.

Distanzberechnung

Die Distanzberechnung, wie sie zu Unterstützung des kartenbasierten Ansatzes in Kapitel 4.2.2 beschrieben wurde, muss alle paarweisen Distanzen zwischen zwei beliebigen Graph-Knoten berechnen, dies ist eine potentiell teuer Operation hinsichtlich der Laufzeit und sollte daher möglichst effizient durchgeführt werden und nach Möglichkeit nur wenige Male im Ablauf der Simulation ausgeführt werden. Um diese Berechnung effizient durchführen zu

können, wird in dieser Arbeit der *Floyd-Warshall-Algorithmus* [Flo62] in einer leicht veränderten Version verwendet. Der Grund für die Veränderung gegenüber dem Originalalgorithmus liegt in der Verbesserung der erwarteten Laufzeit bei n Knoten von $\Theta(n^3)$ auf $\mathcal{O}(n^3)$. Die dafür verwendete Form ist in Algorithmus 5.1 dargestellt. Die Verbesserung der Laufzeit von $\Theta(n^3)$ auf $\mathcal{O}(n^3)$ wird durch die Überprüfung auf $D[i,k] < \infty$ vor der innersten for-Schleife erreicht, dies erspart unnötige Berechnungen, die keine neuen Ergebnisse ergeben würden.

Algorithmus 5.1 Verwendete Form des *Floyd-Warshall-Algorithmus*

```
Eingabe: Adjazenzmatrix A ∈ (ℝ ∪ ∞)_{n×n}
procedure FloydWarshall(A : Adjazenzmatrix)
    D = A                                          // Distanz-Adjazenzmatrix
    for k = 1 to n do
        for i = 1 to n do
            if D[i,k] < ∞ then
                for j = 1 to n do
                    D[i,j] = min{D[i,j], A[i,k] + D[k,j]}
                end for
            end if
        end for
    end for
    return D
end procedure
```

Ein Problem der Distanzberechnung ist dabei deren hohe Komplexität, die zu einer langen Laufzeit führt, wenn n groß wird: Für jede neue Position eines mobilen Knotens und für jeden neu auf dem Server hinzugefügten virtuellen Sensor müssen die Distanzen zu allen anderen Knoten neu berechnet werden, um die Effektivität des Systems zu erhalten. Dies ist allein mit dem vorgestellten Algorithmus nicht effizient durchführbar, da bereits eine Berechnung aller paarweisen Distanzen einige Minuten benötigt, die Berechnung nach jeder Positionserfassung eines mobilen Knoten und bei jedem neuen virtuellen Sensor würde alleine mit diesem Ansatz potentiell einige Jahre für eine einzige Simulation benötigten. Daher wird nun ein verbesserter Ansatz für die Neuberechnung der Distanzen nach einer Positionserfassung oder dem Hinzufügen eines virtuellen Sensors vorgestellt.

Der *Floyd-Warshall-Algorithmus* wird nur zu Beginn der Simulation auf dem gesamten Graphen ausgeführt, alle folgenden Distanzberechnungen basieren dabei darauf, dass bei Positionserfassungen von mobilen Knoten oder neu hinzugefügten virtuellen Sensoren zwar Graph-Knoten und Kanten hinzugefügt werden, diese aber die paarweisen Distanzen der bereits vorhandenen Knoten nicht verändern. Dadurch ist es möglich, die Distanzen nur für die hinzugekommenen Knoten zu berechnen und eine Komplexität bei n Graph-Knoten von $\Theta(n)$ zu erreichen. Aufgrund dieser Beobachtung ist auch das Entfernen von Graph-Knoten, beispielsweise bei Empfang einer neuen Position eines mobilen Knotens oder beim Entfernen eines virtuellen Sensors, kein Komplexitätsproblem, da die paarweisen Distanzen zwischen Graph-Knoten sich nicht verändern und somit keine Neuberechnung der Distanzen vonnöten ist.

Raster-basierte Kantenverwaltung

In Kapitel 4.2.2 wurde das Vorgehen hinsichtlich des Erfassungsbereichs der virtuellen Sensoren und der Ungenauigkeit der Positionserfassung der mobilen Knoten beschrieben, wobei darauf aufbauend die Distanzberechnung zwischen den mobilen Knoten und den virtuellen Sensoren gestartet wird. Um diesen Ansatz umsetzen zu können, muss ein effizienter Ansatz gefunden werden, die durch die Erfassungszonen oder die Kreise der Ungenauigkeit der Positionserfassung geschnittenen Kanten aufzufinden. Die direkte Umsetzung, also eine vollständige Traversierung der Kanten des Graphen für jeden neu erstellten virtuellen Sensor oder für jede neue Position eines mobilen Knoten und ein jeweiliger Test auf Schnittpunkte zwischen der Kante und dem Kreis der Erfassungszone oder dem Kreis der Ungenauigkeit der Positionserfassung, ist zwar effektiv aber nicht effizient und kann potentiell die Laufzeit der Simulation stark erhöhen. Daher soll nun ein verbesserter Ansatz zur Detektion von Kanten in der Umgebung eines virtuellen Sensors oder mobilen Knotens vorgestellt werden.

Für die vorliegende Arbeit wird ein Raster-basierter Ansatz gewählt, dieser beruht auf dem Prinzip, dass bei ausreichender Maschenweite (hier der größte Radius aller virtuellen Sensoren) nur die Kanten von 9 Planquadraten untersucht werden müssen, siehe Grafik 5.2. Dieses Vorgehen entspricht einer deutlichen Verbesserung gegenüber einem naiven Ansatz mittels Traversierung des kompletten Graphen und ermöglicht somit eine effiziente Durchführung der Simulation.

Graph Verwaltung Nachdem der Raster-basierte Ansatz eingeführt wurde, soll nun gezeigt werden, wie die Verwaltung des Graphen und des Rasters beim Hinzufügen eines virtuellen

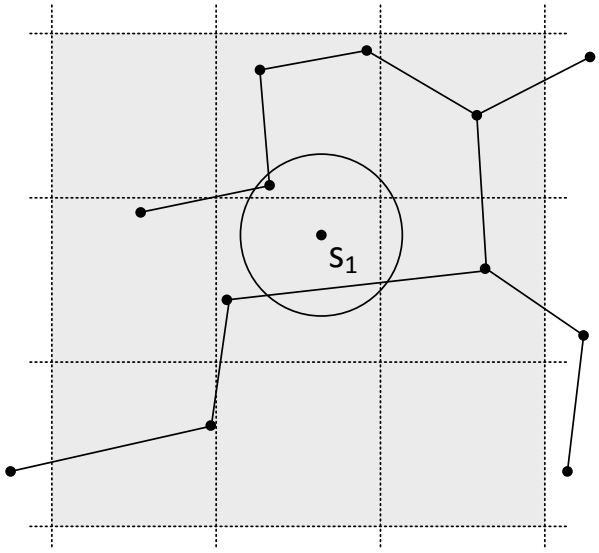


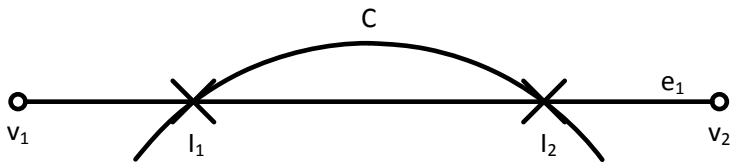
Abbildung 5.2: Visualisierung der möglichen Nachbarquadrate, die möglicherweise die Erfassungszone von s_1 schneiden

Sensors oder bei einer neuen Position eines mobilen Knotens abläuft:
Kanten, die komplett innerhalb des betrachteten Kreises der Erfassungszone oder der Positionserfassungsunsicherheit liegen, werden genauso wenig weiter betrachtet wie Kanten, die komplett außerhalb des Kreises liegen, einzig scheidende oder tangierende Kanten werden weiterverarbeitet: Deren Schnittpunkte mit dem betrachteten Kreis werden berechnet und für die weitere Verarbeitung zwischengespeichert. Danach werden die berechneten Schnittpunkte in Graph-Knoten umgewandelt und dem Graphen hinzugefügt, die bereits bestehende Kante wird entfernt und neue Kanten dem Graphen hinzugefügt, Grafik 5.3 zeigt dabei das Vorgehen bei zwei Schnittpunkten. Das Vorgehen wird mittels Pseudocode auch durch Algorithmus 5.2 beschrieben.

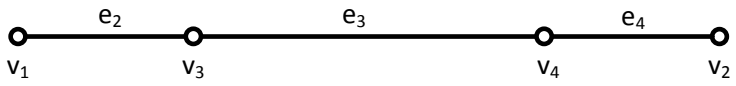
Algorithmus 5.2 Algorithmus zur Verarbeitung der Schnittpunkte

```
procedure AddNodesAndEdges(nachbarKanten, Kreis)
    for all nachbarKanten as Kante do
        if Kreis schneidet Kante ∨ Kreis tangiert Kante then
            entferne Kante aus Graph und Raster
            for all Schnittpunkte zwischen Kreis und Kante do
                füge neue Knoten für Schnittpunkt in Graph ein
            end for
            if genau 1 Schnittpunkt zwischen Kreis und Kante then
                verbinde neuen Knoten mit Endpunkten von Kante
            else
                verbinde jeweils näherliegenden Knoten mit Endpunkten von Kante
                verbinde neue Knoten
            end if
            füge neue Kanten in Graph und Raster ein
        end if
    end for
end procedure
```

Auffinden überstrichener Planquadrate Um das oben beschriebene Vorgehen mittels Raster zur Entdeckung nächstgelegener Kanten auch praktisch verwenden zu können, muss nun ein Verfahren gefunden werden, die von einer Kante überstrichenen Planquadrate des Rasters möglichst effizient aufzufinden. Dies ist dabei von umso größerer Bedeutung, je mehr mobile Knoten an der Simulation teilnehmen, da diese bei jeder Positionserfassung neue Kanten dem Graph hinzufügen und alte Kanten aus dem Graph löschen, diese Kanten müssen auch im Raster hinzugefügt und gelöscht werden. Wird zum Auffinden der überstrichenen Planquadrate nun ein naiver Algorithmus verwendet, der jedes Planquadrat des Rasters durchläuft und jeweils überprüft, ob die entsprechende Kante innerhalb dieses Quadrats liegt, so resultiert daraus eine Laufzeit mit quadratischem Zusammenhang zur Breite und Länge des Rasters. Damit skaliert auch die Simulation nur quadratisch in der Anzahl der teilnehmenden mobilen Knoten wie auch in der Zahl der hinzugefügten virtuellen Sensoren, das Ziel ist aber ein möglichst linearer Zusammenhang zwischen der Simulationszeit und



(a) Die Ausgangssituation Kreis C der die Kante e_1 schneidet



(b) Situation nach Ausführung des Schnittes mit neuen Knoten v_3, v_4 ohne die Kante e_1 und den neuen Kanten e_2, e_3, e_4

Abbildung 5.3: Visualisierung der Veränderung des Graphen durch neue Knoten und Kanten, aus der Situation in Grafik 5.3a entsteht dabei die Situation in Grafik 5.3b

der Anzahl der mobilen Knoten und virtuellen Sensoren, weswegen nun ein verbesserter Ansatz zur Berechnung der von einer Kante überstrichenen Planquadrate vorgestellt werden soll.

Zur Berechnung der von einer Kante überstrichenen Planquadrate wird eine abgewandelte Version von *Bresenham's Line Algorithm* [Bre65] verwendet: In der Standardversion dieses Algorithmus werden nicht alle überstrichenen Planquadrate ausgegeben, da der originale Algorithmus für die Approximation von Linien mittels Planquadraten entworfen wurde, daher wird der Algorithmus im Rahmen dieser Arbeit für die Ausgabe aller überstrichener Planquadrate modifiziert, eine ähnliche Modifikation ist auch in [Coh] für den dreidimensionalen Fall dargestellt. Der Algorithmus wird zudem zusätzlich um die Fähigkeit erweitert, abgesehen von Einheitsquadraten auch beliebige Seitenlängen der Planquadrate verarbeiten zu können. Der Pseudocode für eine Linie von *start* nach *ende* ist in Algorithmus 5.3 dargestellt.

Algorithmus 5.3 Modifizierte Version von *Bresenham's Line Algorithm*

```
procedure BresenhamModifiziert(start, ende)
    Starte mit Planquadrat unter start
    for Anzahl an geschnittenen Planquadraten do
        Registriere aktuelles Planquadrat
        if Nächter Schnittpunkt ist mit horizontaler Rasterlinie then
            Bewegung in horizontaler Richtung in Richtung von ende
        else
            Bewegung in vertikaler Richtung in Richtung von ende
        end if
    end for
end procedure
```

Dieser einfache Algorithmus kann dann mit folgenden Zusatzinformationen ausgeführt werden:

Anzahl der geschnittenen Planquadrate Die Anzahl an geschnittenen Planquadraten lässt sich berechnen durch:

(5.1) $1 + |\text{Schnitte mit vertikalen Rasterlinien}| + |\text{Schnitte mit horizontalen Rasterlinien}|$

wobei die 1 das Startfeld symbolisiert, da das Startfeld auch ohne Schnitt mit einer Rasterlinie ausgegeben werden soll.

Bestimmung horizontaler und vertikaler Schnittpunkte Auf den ersten Blick schwieriger scheint die Bestimmung, ob der nächste Schnittpunkt mit einer horizontaler oder vertikaler Rasterlinie erfolgt. Auch dieses Problem kann leicht gelöst werden, indem über den nächsten Schnittpunkt in horizontaler und vertikaler Richtung Buch geführt wird. Dies geschieht dabei in Abhängigkeit der bereits zurückgelegten Strecke im Vergleich zur gesamten Linie, dies lässt sich auf die Grundform einer Geraden g als $g = \vec{r} + t \cdot \vec{v}$ mit Ortsvektor $\vec{r}$ und Richtungsvektor $\vec{v}$ zurückführen.

Beispiel Das folgende Beispiel erläutert das Vorgehen, die Grafik 5.4 visualisiert dabei das Beispiel:
Sei $start = a = (1,1)$ und $ende = b = (3,7)$ auf dem 8×8 Gitter mit Maschenweite 2: Der erste (und einzige) Schnittpunkt mit einer vertikalen Rasterlinie befindet sich bei $t = \frac{1}{2}$, die Schnittpunkte mit horizontalen Rasterlinien finden sich bei $t = \frac{1}{6}, t = \frac{3}{6}, t = \frac{5}{6}$. Es liegen also nach Gleichung (5.1) insgesamt $1 + 1 + 3 = 5$ überstrichene Planquadrate vor, gestartet wird nach dem Algorithmus in Planquadrat $(0,0)$, danach wird ein Planquadrat in vertikaler Richtung in Richtung von *ende* besucht, also $(0,1)$, danach folgen die Planquadrate $(1,1)$, $(1,2)$ und $(1,3)$. Damit sind alle Planquadrate gefunden, welche die Linie von *start* nach *ende* durchschneidet.

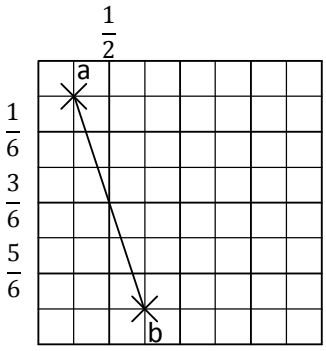


Abbildung 5.4: 8×8 Raster mit Maschenweite 2, die betrachtete Linie führt dabei von a nach b

Mithilfe dieses Algorithmus können also die von einer Kante überstrichenen Planquadrate effizient gefunden werden, es werden immer nur die Planquadrate besucht, welche die Kante auch schneidet. Dies stellt eine enorme Verbesserung gegenüber einem naivem Ansatz dar, welcher alle Planquadrate auf eventuelle Schnittpunkte mit der betrachteten Kante untersucht, dieser Ansatz hat bei einer Kantenlänge des Gitters von n Einträgen eine Komplexität von $\Theta(n^2)$. Der vorgestellte Algorithmus hat im Gegensatz dazu eine Komplexität von $\mathcal{O}(n)$, der Worst-Case ist dabei eine Kante von $(1,1)$ nach (n,n), dabei werden $\sqrt{2} \cdot n$ Planquadrate besucht.

6 Evaluation

Das folgende Kapitel widmet sich der Evaluation des in den Kapiteln 4 und 5 vorgestellten Public Sensing-Systems. Dabei ist anzumerken, dass sich die Ergebnisse alle auf Simulationen beziehen, da es aufgrund der großen Anzahl an unterschiedlichen Szenarien und der hohen Teilnehmerzahl, nicht möglich war, dass System in einer realen Umgebung zu evaluieren.

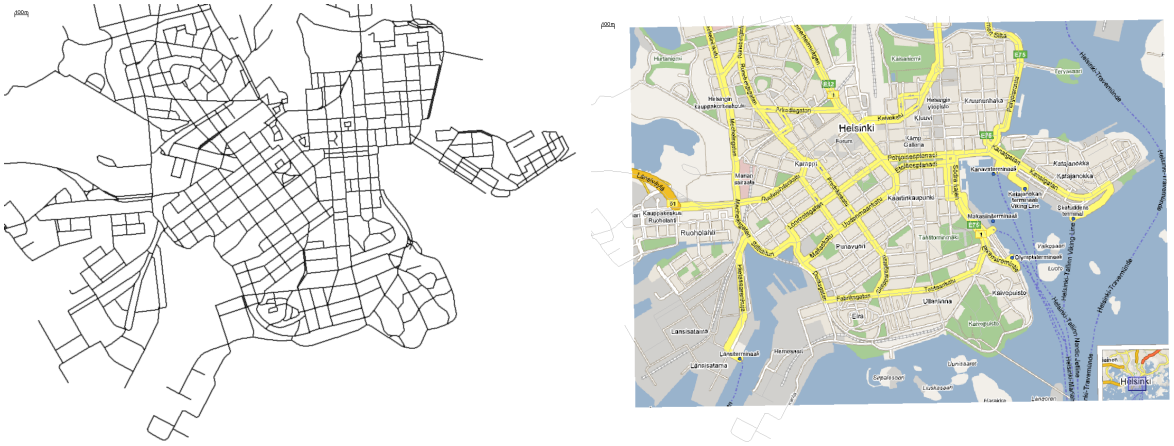
6.1 Parameter

Das entwickelte System soll nun mittels mehrerer, verschiedener Szenarien evaluiert werden, dazu müssen die Parameter des Systems definiert werden. Dabei gibt es mehrere Arten von Parametern: Es gibt globale Parameter, welche die Simulation an sich betreffen, es gibt Parameter, welche die Arbeitsweise des Servers beeinflussen, und nicht zuletzt gibt es Parameter zur Beeinflussung der mobilen Knoten.

6.1.1 Globale Parameter

Ein globaler Parameter im Zusammenhang mit der durchzuführenden Simulation ist das zugrundeliegende Simulationsgebiet. In den folgenden Auswertungen wird dabei stets dasselbe Gebiet verwendet, der Graph ist dabei in Grafik 6.1 abgebildet. Ein weiterer Parameter ist die Bewegung der mobilen Knoten auf dem zur Verfügung stehenden Gebiet, diese wurde für die folgenden Experimente auf eine pseudorandomisierte Bewegung gesetzt, um die Wiederholbarkeit der Experimente zu garantieren. Die Experimente wurden dabei mit verschiedenen Anzahlen von teilnehmenden mobilen Knoten zwischen 50 und 2000 durchgeführt.

Für die Erzeugung virtueller Sensoren müssen des Weiteren die Parameter k (die Mindestanzahl an Werten, bevor der virtuelle Sensor als erfasst gilt) und t_1, die Zeitmarke, ab welcher der virtuelle Sensor als verpasst gilt, gesetzt werden, diese werden keiner der beiden folgenden Untersektionen zugeordnet, sondern symbolisieren globale Parameter. Für alle folgenden Simulationen wurde k auf 1 und t_1 auf 600 $s = 10$ min gesetzt.



(a) Darstellung des reinen Graphen der Simulation

(b) Darstellung des Gebietes der Simulation: die Innenstadt von Helsinki auf einer Google Maps-Karte

Abbildung 6.1: Graph des Simulationsgebiets

6.1.2 Server

Die Parameter des Servers bestehen aus zwei Gruppen: zum einen kann für Art der Verteilung der virtuellen Sensoren zwischen mehreren Varianten gewählt werden, namentlich zwischen der adaptiven Verteilung und dem naiven Ansatz sowie als weitere Option für die adaptive Verteilung zudem der kartenbasierte Verteilungsansatz. Die zweite Gruppe von Parameter bezieht sich auf die Generierung der virtuellen Sensoren, diese werden an zufälligen Positionen generiert wobei die maximale Anzahl der virtuellen Sensoren festgelegt werden kann sowie das Intervall zwischen zwei generierten virtuellen Sensoren angegeben werden kann.
Damit stehen für den Server drei verschiedene Ansätze zur Verteilung der virtuellen Sensoren zur Verfügung: Ein naiver Ansatz, ein adaptiver Ansatz ohne Einbringung des Wissens über den zugrunde liegenden Graphen und ein kartenbasierter adaptiver Ansatz mit Verwendung des Graphen. Diese Ansätze sollen nun in Kapitel 6.2 verglichen werden.

6.1.3 Mobile Knoten

Die Parameter der mobilen Knoten betreffen vor allem die Steuerung der Parameter des simulierten GPS. Das in Kapitel 4.3 vorgestellte Adaptive GPS kann aktiviert werden, wird dieses nicht aktiviert, so kann für das normale GPS ein Intervall zwischen zwei Positionserfassungen vorgegeben werden. Zudem können die in Kapitel 4.1.1 einzeln aktiviert oder deaktiviert werden wobei für jedes der Update-Protokolle die spezifischen Parameter gesetzt werden können. Zudem muss die maximale Geschwindigkeit eines mobilen Knotens angegeben werden, diese ist für die korrekte Funktionsweise des adaptiven GPS von großer

Bedeutung. Für die mobilen Knoten werden in Kapitel 6.2 damit zum einen das adaptive GPS von Bedeutung und zum anderen das verwendet Update-Protokoll.

6.1.4 Überblick

Die folgende Liste gibt einen Überblick über die verschiedenen Parameter des simulierten Systems:

- Globale Parameter:
  - Anzahl der teilnehmenden mobilen Knoten
  - Gebiet der Simulation
- Parameter der virtuellen Sensoren:
  - Mindestanzahl an Erfassungen $k = 1$
  - Verpasst-Zeitmarke $t_1 = t_0 + 600\ s$
- Parameter des Servers:
  - naive Verteilung der virtuellen Sensoren
  - adaptive Verteilung der virtuellen Sensoren mit euklidischen Distanzen
  - kartenbasierte adaptive Verteilung der virtuellen Sensoren
- Parameter der mobilen Knoten:
  - Herkömmliche Positionserfassung
  - Adaptive Positionserfassung
  - Aktivierte Update-Protokolle mit jeweiligen Parametern

6.2 Auswertung

Das folgende Kapitel beschäftigt sich mit der Evaluation, es werden dabei verschiedene Szenarien vorgestellt wobei diese mittels Variation der vorgestellten Parameter erzeugt werden. Dazu wird im ersten Schritt die Effektivität des Systems mit verschiedenen Ausgangsparametern verglichen um für die folgenden Auswertungen die Sicherheit zu haben, dass die verschiedenen Systeme bezüglich der Effektivität äquivalent sind. Danach soll zuerst die Anzahl der versendeten virtuellen Sensoren verglichen werden, gefolgt von der Diskussion des Einflusses des adaptiven GPS. Anschließend wird der Einfluss der Verwendung eines Update-Protokolls auf den mobilen Knoten diskutiert, bevor die Evaluation des Energieaufwands die Auswertung beschließt.

6.2.1 Effektivität

Zunächst sollen die verschiedenen im Zusammenhang mit der Anzahl der erfassten virtuellen Sensoren verglichen werden. Die verschiedenen Ansätze unterscheiden sich geringfügig in der Zahl der erfassten virtuellen Sensoren, dies lässt sich durch abweichende Positionserfassungsungenauigkeiten erklären, da für zwei nahe beieinander liegende Positionserfassungen auch größere Abweichungen möglich sind, somit ist eine Erfassung eines virtuellen Sensors in einem Fall möglich und im anderen nicht. Zudem ist es möglich, dass sich Knoten nicht auf einer Geraden durch die Erfassungszone eines virtuellen Sensors bewegen, was das in Kapitel 4.1.3 vorgestellte Verfahren abdecken kann, sondern der Pfad des mobilen Knotens einen Linienzug beschreibt, der eine Erfassung mittels des vorgestellten Verfahrens unmöglich macht, Grafik 6.2 visualisiert diese Situation.

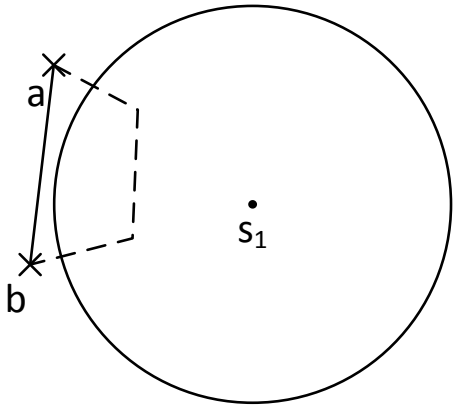


Abbildung 6.2: Visualisierung der Situation einer nicht vorgenommenen Erfassung aufgrund eines durch Positionserfassungen nicht registrierten Pfades des mobilen Knotens durch die Erfassungszone von s_1, an den Positionen a und b erfolgt dabei eine Positionserfassung, die Linie durch a und b schneidet die Erfassungszone von s_1 nicht, eine Erfassung kann nicht erkannt werden

Die folgenden Grafiken zeigen nun einen Vergleich zwischen der Anzahl der erfassten virtuellen Sensoren und der Anzahl an der Simulation teilnehmender mobiler Knoten. Grafik 6.3b stellt diesen Vergleich mit Verwendung des adaptiven Positionserfassung dar, Grafik 6.3a ohne Verwendung der adaptiven Positionserfassung. Die Unterschiede zwischen den verschiedenen Strategien lassen sich dabei durch oben dargelegte Effekte erklären, wobei die verschiedenen Ansätze bis auf eine geringe, erklärbare Abweichung die gleiche Effektivität aufweisen und somit eine Grundvoraussetzung für die verbesserten Ansätze, also eine mindestens gleichwertige Effektivität wie im naiven Ansatz, gegeben ist. Des Weiteren zeigen dieser Vergleich der Effektivität gut den Einfluss einer größeren Anzahl von teilnehmenden mobilen Knoten: Mit zunehmender Anzahl an mobilen Knoten werden zwar auch mehr virtuelle Sensoren erfasst, der Anstieg erfolgt dabei aber nicht linear, sondern nähert sich einem Grenzwert an. Dies ist dadurch erklärbar, dass zum einen nur eine begrenze Anzahl an virtuellen Sensoren generiert wird, wobei durch die größere Anzahl an teilnehmenden

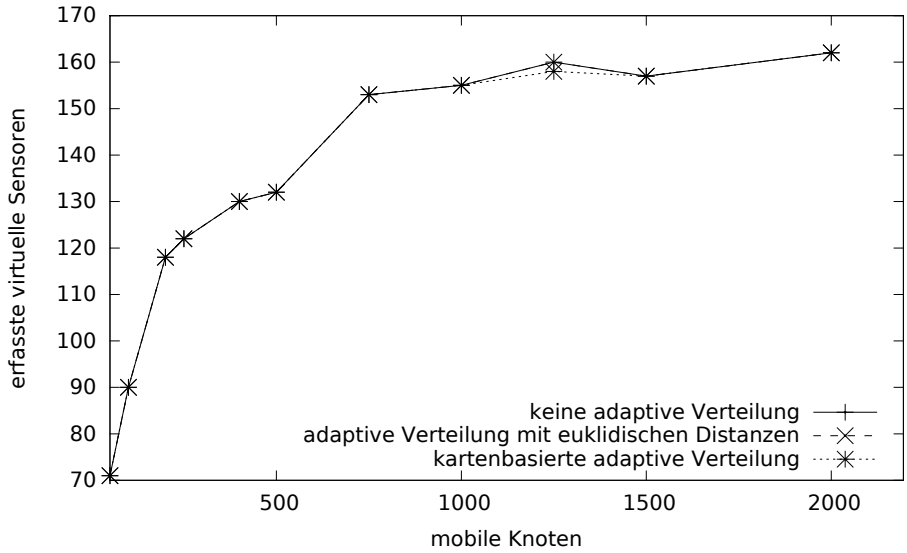
mobilen Knoten auch virtuelle Sensoren in Gebieten abseits des Zentrums erfasst werden, wobei eine Erfassung aller virtueller Sensoren durch das relativ weitläufige Gebiet mit vielen möglichen Pfaden unwahrscheinlich ist.

6.2.2 Vergleich verteilte virtuelle Sensoren

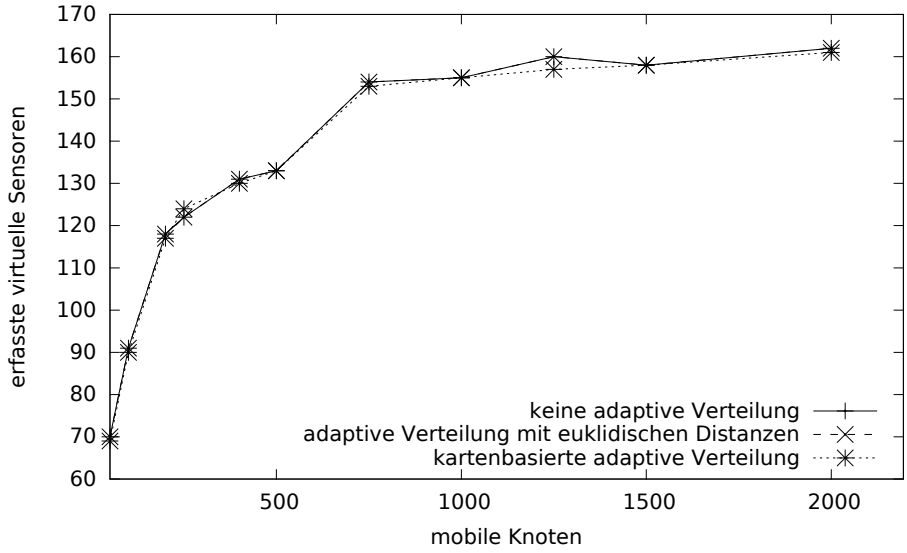
Da wie gezeigt der naive Ansatz und die optimierten Ansätze hinsichtlich der Effektivität vergleichbar sind, soll nun ein Vergleich der Anzahl der auf die mobilen Knoten verteilten virtuellen Sensoren zeigen, inwiefern die optimierten Ansätze eine Verbesserung der Anzahl der verteilten virtuellen Sensoren ergibt. Dazu wird das von den mobilen Knoten verwendete Update-Protokoll auf das Simple Protokoll (siehe Kapitel 4.1.1) festgelegt, um diesen Einfluss auf die Anzahl der erfassten virtuellen Sensoren von vornherein auszuschalten, dieser soll später in Kapitel 6.2.4 diskutiert werden.
Wie Grafik 6.4 zeigt, ist die Anzahl an verteilten virtuellen Sensoren bei Verwendung einer adaptiven Verteilungsstrategie um ein Vielfaches geringer als bei dem naiven Ansatz. Da Details zu den adaptiven Ansätzen nicht sichtbar werden, befasst sich der folgende Abschnitt mit einer genaueren Analyse der adaptiven Verteilung.

Vergleich euklidische Distanzen zu kartenbasiertem Ansatz

Nach der vorangegangenen Analyse über alle Verteilungsansätze sollen nun die beiden adaptiven Ansätze, die Berechnung der Distanz mittels der Verwendung der euklidischen Distanz $\|\cdot\|_2$ und die Berechnung der Distanzen mittels kartenbasiertem Ansatz verglichen werden. Als weiteres Vergleichsmerkmal soll nun auch die Anwendung des adaptiven GPS auf den mobilen Knoten zur Anwendung kommen, um den Einfluss dieser Kenngröße auf die Anzahl der verteilten virtuellen Sensoren zu ermitteln. Grafik 6.5 zeigt die Ergebnisse auf, auffällig ist der große Unterschied an verteilten virtuellen Sensoren zwischen der Verwendung des adaptiven GPS und der Verwendung des normalen GPS sowohl bei der Anwendung euklidischen Distanzen als auch bei Anwendung des kartenbasierten Ansatzes. Dies ist erklärbar durch die Funktionsweise des adaptiven GPS im Zusammenspiel mit der adaptiven Verteilung : die Verwendung des adaptiven GPS verringert die Anzahl der GPS-Positionserfassungen um ein Vielfach. Dies bedeutet aber für die Verfahren der adaptiven Verteilung der virtuellen Sensoren, dass die Unsicherheit der Position größer wird als bei der Verwendung des normalen GPS, diese resultiert schlussendlich in eine größerer Anzahl an verteilten virtuellen Sensoren. Die Grafik 6.5b zeigt dabei eine Detailansicht der Grafik 6.5a, womit deutlich wird, wie effizient der Ansatz der adaptiven Verteilung der virtuellen Sensoren ist: Statt 358000 verteilten virtuellen Sensoren bei der Verwendung der naiven Verteilung und 2000 teilnehmenden mobilen Knoten werden nun bei Verwendung des kartenbasierten Ansatzes bei gleicher Teilnehmeranzahl nur ca. 3000 virtuelle Sensoren an die mobilen Knoten verteilt, bei äquivalenter Effektivität entspricht dies nun einer Reduzierung der Anzahl verteilten virtuellen Sensoren um $\approx 99\%$. Auch ist die Verbesserung des kartenbasierten Ansatzes deutlich zu erkennen, der $\approx 20\%$ weniger virtuelle Sensoren an mobile Knoten verteilt.

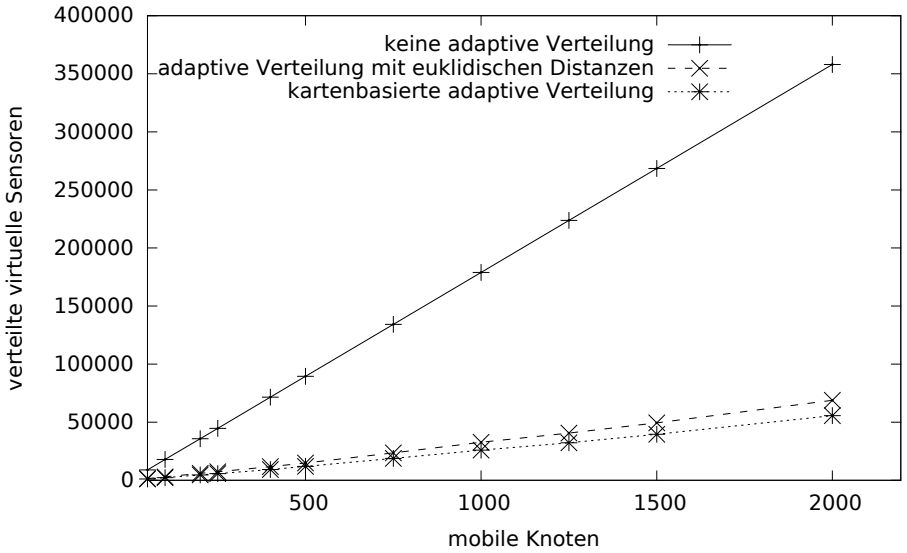


(a) erfasste virtuelle Sensoren ohne Verwendung der adaptiven Positionserfassung

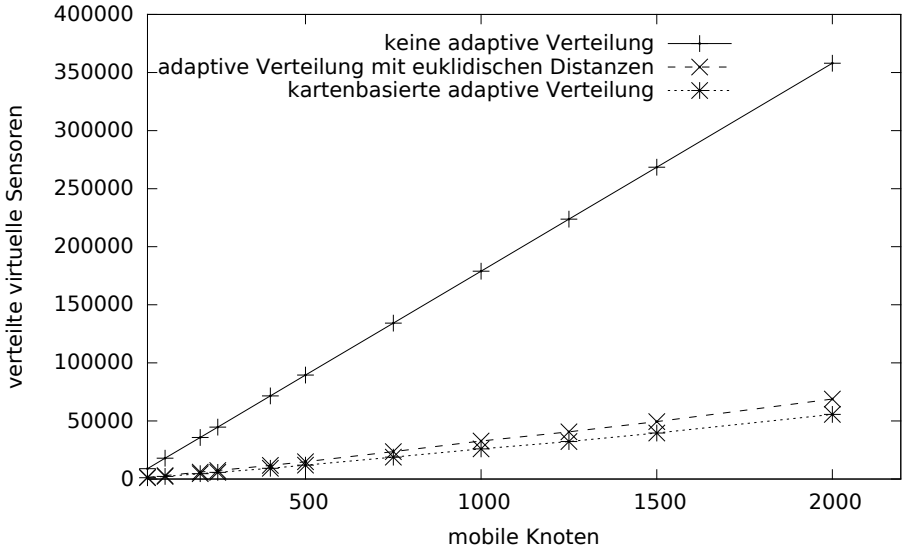


(b) erfasste virtuelle Sensoren mit Verwendung der adaptiven Positionserfassung

Abbildung 6.3: Vergleich der Anzahl erfasster virtueller Sensoren ohne und mit Verwendung der adaptiven Positionserfassung

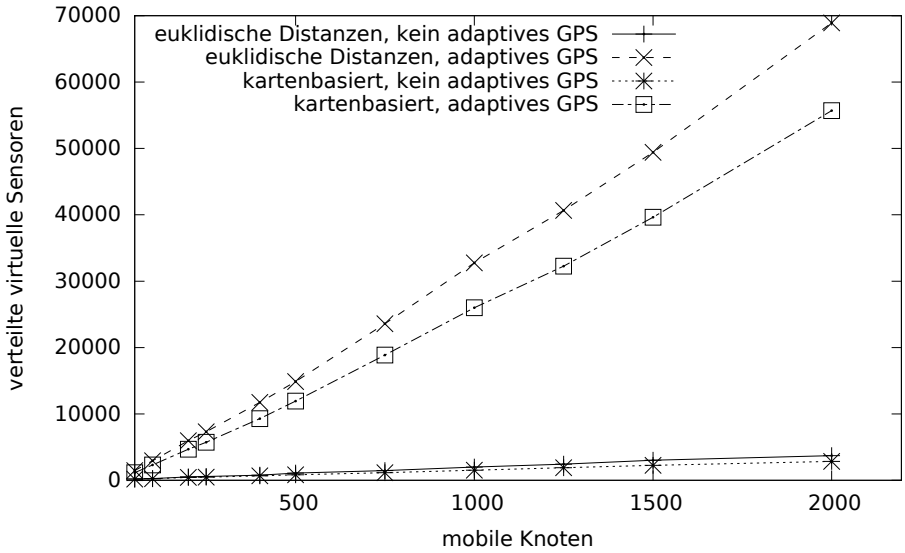


(a) verteilte virtuelle Sensoren ohne Verwendung der adaptiven Positionserfassung

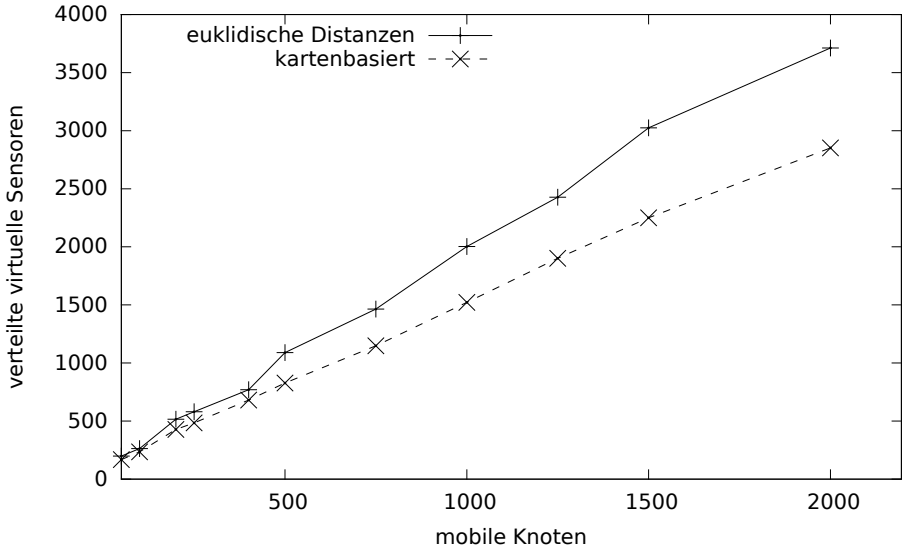


(b) verteilte virtuelle Sensoren mit Verwendung der adaptiven Positionserfassung

Abbildung 6.4: Vergleich der Anzahl verteilter virtueller Sensoren jeweils ohne Verwendung eines Update-Protokolls



(a) verteilte virtuelle Sensoren mit Verwendung der adaptiven Verteilung



(b) Detailansicht der Grafik 6.5a ohne Verwendung der adaptiven Positionserfassung

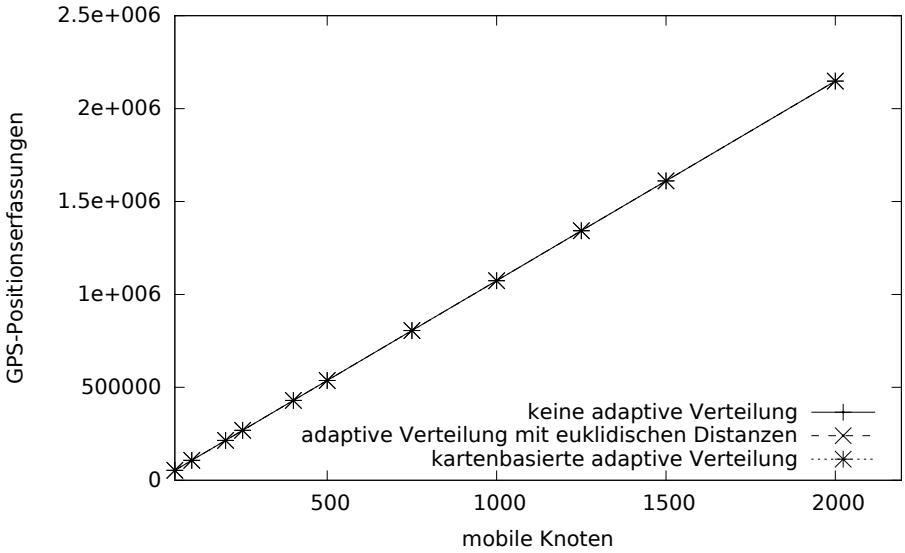
Abbildung 6.5: Vergleich der Anzahl verteilter virtueller Sensoren mit Verwendung der adaptiven Verteilung, jeweils ohne Verwendung eines Update-Protokolls

6.2.3 Vergleich GPS-Positionserfassungen

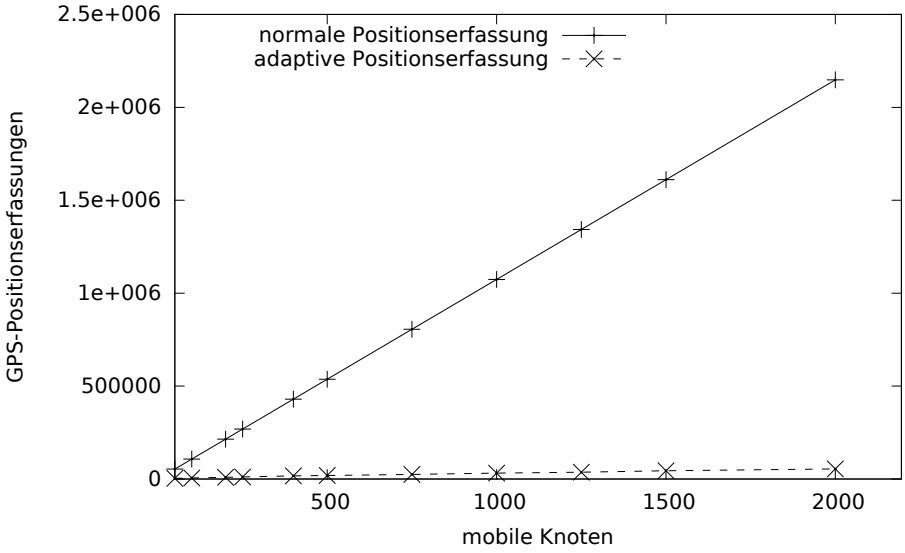
Nachdem der Einfluss der adaptiven Verteilung auf die Anzahl der virtuellen Sensoren diskutiert wurde, soll nun der Einfluss auf die Anzahl der GPS-Positionserfassungen diskutiert werden, dabei soll vor allem der potentielle Vorteil des adaptiven GPS diskutiert werden. Dabei soll auch hier kein Update-Protokoll zum Einsatz kommen, um den direkten Einfluss der Positionserfassungsstrategie vergleichen zu können. Die Grafik 6.6 zeigt dazu die Ergebnisse, wobei in Grafik 6.6a die Ergebnisse ohne Verwendung der adaptiven Positionserfassung abgetragen sind. Wie nicht anders zu erwarten hat hier eine Variation der Verteilungsstrategie keinen Einfluss auf die Anzahl der Positionserfassungen, da das normale GPS eine feste Update-Rate vorgibt und diese ohne Einfluss der Verteilungsstrategie einhält. Um die Vergleichbarkeit der verschiedenen Ansätze zu gewährleisten, enthält Grafik 6.6b den direkten Vergleich der herkömmlichen Positionserfassung gegenüber der adaptiven Positionserfassung, jeweils mit dem grundlegenden System der Verteilung der virtuellen Sensoren, dabei wird die verringerte Anzahl an Positionserfassungen deutlich.
Auch die verschiedenen Verteilungsstrategien haben bei Verwendung der adaptiven Positionserfassung einen Einfluss, dies ist in Grafik 6.7a sichtbar: Hier wird vor allem durch die Verwendung der kartenbasierten adaptiven Verteilung eine Verringerung der Positionserfassungen deutlich. Dies ist damit zu erklären, dass durch die Verwendung der kartenbasierten adaptiven Verteilung eine verringerte Anzahl an virtuellen Sensoren an die mobilen Knoten versendet werden, wodurch diese wiederum mittels der adaptiven Positionserfassung weniger GPS-Positionserfassungen vornehmen müssen. Ein weiterer Effekt zeigt sich bei der Betrachtung der Positionserfassungen pro mobilem Knoten in Grafik 6.7b: Je mehr mobile Knoten teilnehmen, desto weniger Positionserfassungen werden pro Knoten vorgenommen. Dies hat als Ursache die bessere Abdeckung des Gebiets mit mehr mobilen Knoten und damit auch eine schnellere Erfassung der virtuellen Sensoren bei mehr teilnehmenden Knoten. Dies hat wiederum für die adaptive Positionserfassung den Vorteil, dass die Distanz zum nächsten erreichbaren Knoten bei einer geringeren Anzahl an aktiven virtuellen Sensoren tendenziell größer ist und somit eine längere Phase der Deaktivierung des GPS bis zur nächsten Positionserfassung erlaubt, bei jeweils gleichem Simulationszeitraum werden also mit mehr teilnehmenden Konten weniger Positionserfassungen pro Konten durchgeführt. Wie schon in Kapitel 6.2.2 gezeigt werden durch die geringere Anzahl an Positionserfassungen größere Positionsunsicherheiten für die adaptiven Verteilungsstrategien erzeugt, was wiederum eine größere Anzahl an versendeten virtuellen Sensoren bedingt. Die Auswirkungen beider Einzelparameter auf den Energieaufwand und damit auf das schlussendliche Minimierungsziel werden in Kapitel 6.2.5 beschrieben.

6.2.4 Vergleich Update-Protokolle

Nachdem sich die bisherigen Analysen lediglich mit dem Simple Update-Protokoll und damit de-facto keinem Update-Protokoll befassten und andere Bereiche wie die Anzahl der GPS-Positionserfassungen oder die Anzahl an verteilten virtuellen Sensoren untersuchten soll nun auch der Einfluss der anderen Update-Protokolle untersucht werden. Dazu werden verschiedene Update-Protokolle hinsichtlich ihres Einflusses auf die Anzahl der verteilten

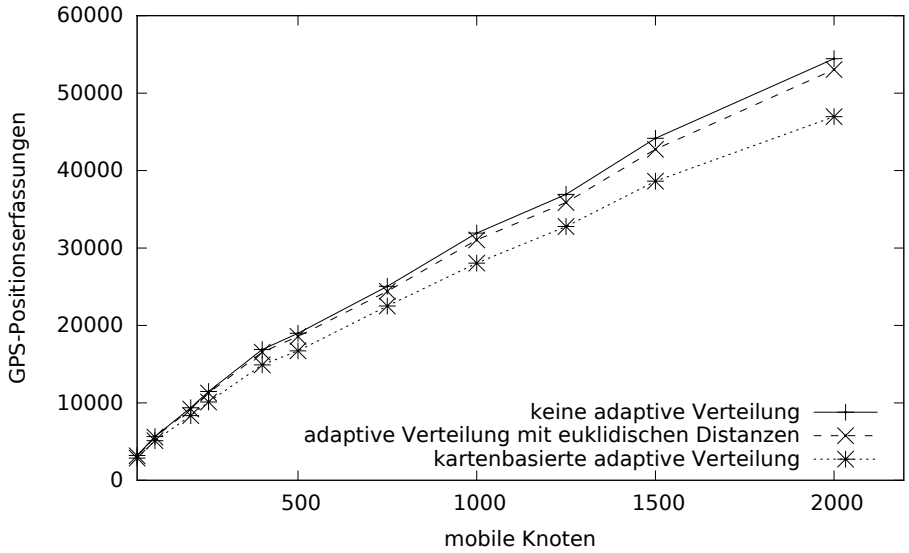


(a) Anzahl der GPS-Positionserfassungen ohne adaptive Positionserfassungsstrategie

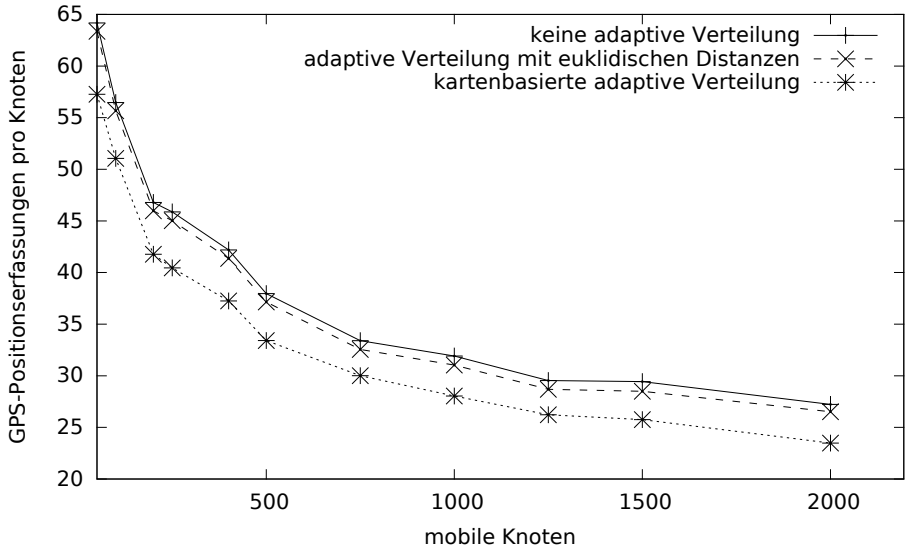


(b) Vergleich der Anzahl der GPS-Positionserfassungen hinsichtlich der Positionserfassungsstrategie, ohne adaptive Verteilung der virtuellen Sensoren

Abbildung 6.6: Vergleich der Anzahl der GPS-Positionserfassungen auf den mobilen Knoten mit Vergleich der normalen Positionserfassung zur adaptiven Positionserfassung



(a) Anzahl der GPS-Positionserfassungen mit adaptiver Positionserfassungsstrategie



(b) Vergleich der Anzahl der GPS-Positionserfassungen pro mobilem Knoten mit verschiedenen Verteilungsstrategien und ohne Update-Protokoll

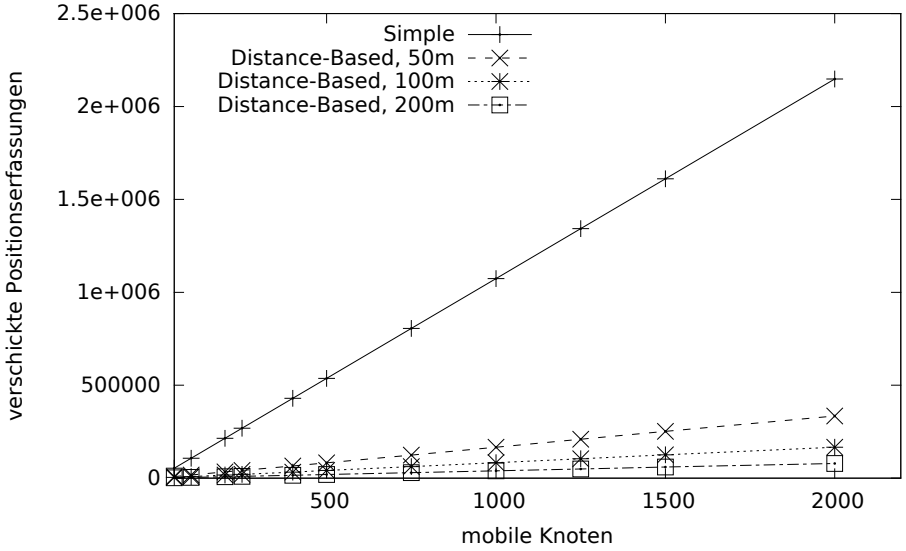
Abbildung 6.7: Vergleich der Anzahl der GPS-Positionserfassungen auf den mobilen Knoten mit adaptiver Positionserfassung und verschiedenen Strategien zur Verteilung der virtuellen Sensoren

virtuellen Sensoren und die Anzahl der an den Server gesendeten GPS-Positionserfassungen untersucht. Zusätzlich wird bei der adaptiven Verteilung der virtuellen Sensoren und der adaptiven Positionserfassung der mobilen Knoten auch der Einfluss eines Update-Protokolls auf die Anzahl der reine GPS-Positionserfassungen untersucht, da diese bei der adaptiven Positionserfassung tendenziell von der Anzahl der bekannten virtuellen Sensoren abhängt welche bei verschiedenen Update-Protokollen potentiell differiert.

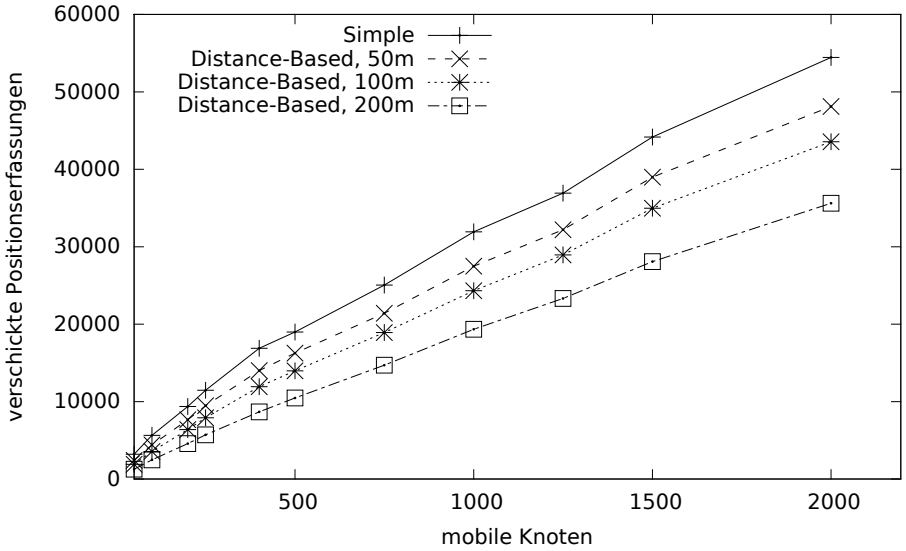
Vergleich gesendete GPS-Positionserfassungen

Zuerst soll nun die Anzahl der an den Server gesendeten Positionserfassungen verglichen werden, dabei wird für die Verteilung der virtuellen Sensoren die naive Strategie angewandt und die mobilen Knoten nutzen die normalen GPS-Positionserfassung, Grafik 6.8 zeigt dazu die Ergebnisse. Wie deutlich zu erkennen ist, verringert die Verwendung das Distance-based Update-Protokolls die Anzahl der versendeten Positionserfassungen drastisch. Die Grafik 6.8b zeigt zusätzlich die Ergebnisse mit Verwendung der adaptiven Positionserfassung. Auffällig ist dabei die im Vergleich zur normalen Positionserfassung deutlich verringerte Anzahl an versendeten Positionsinformationen, welche sich mit der deutlich geringeren Anzahl an Positionserfassungen erklären lässt. Des Weiteren ist anzumerken, dass die Zahl der versendeten Positionserfassungen im Fall der normalen GPS-Positionserfassung und einer Mindestabweichung von 200 m immer noch mehr Positionen versendet als das Simple Protokoll im Fall der adaptiven Positionserfassung, womit der Vorteil der adaptiven Positionserfassung zum wiederholten Male zu Tage tritt.

Eine Untersuchung des Time-based Update-Protokolls mit verschiedenen Parametern ergibt das erwartete Ergebnis: Je größer die Zeit zwischen zwei versendeten Positionserfassungen desto kleiner die Anzahl der versendeten Positionserfassungen. Grafik 6.9a bildet deswegen einen Vergleich des Time-based mit dem Distance-based Update-Protokoll ab, wobei deutlich wird, dass das Distance-based Protokoll eine deutliche Verbesserung der Anzahl der versendeten Positionserfassungen gegenüber dem Time-based Protokoll erbringt, da vor allem bei der 200 m-Einstellung des Distance-based Protokolls sich die mobilen Knoten länger innerhalb dieser Zone aufhalten und damit weniger Positionserfassungen versenden als bei einer festen Rate von 60 s des Time-based Protokolls.
Zur Einordnung eines kombinierten Update-Protokolls zeigt Grafik 6.9b eine Übersicht über die versendeten Positionserfassungen bei Verwendung der Kombination des Time-based und Distance-based Update-Protokolls, dabei fällt auf, dass sich im Vergleich zum normalen Time-based Protokoll mit gleichem Zeit-Parameter eine minimal vermehrte Anzahl an versendeten Positionsinformationen ergibt. Dies hat seinen Ursprung darin, dass zwar bei der Verwendung des kombinierten Protokolls beide Update-Protokolle einer Übertragung der Positionsinformation zustimmen müssen und damit eigentlich weniger Positionsinformationen versendet werden müssten, dieser Effekt aber durch die Verwendung der adaptiven Positionserfassung und der adaptiven Verteilung der virtuellen Sensoren aufgehoben wird: Dadurch, dass weniger Positionsinformationen gesendet werden, werden dem mobilen Knoten eine größere Anzahl an virtuellen Sensoren mitgeteilt, worauf dieser wiederum potentiell eine kürzere Deaktivierungsphase der adaptiven Positionserfassung veranschlagen



(a) versendete Positionsinformationen ohne adaptiver Positionserfassung



(b) versendete Positionsinformationen mit adaptiver Positionserfassung

Abbildung 6.8: Vergleich der Anzahl versendeter Positionsinformationen an den Server mit und ohne adaptive Positionserfassung und mit jeweils verschiedenen Einstellungen des Distance-based Update-Protokolls

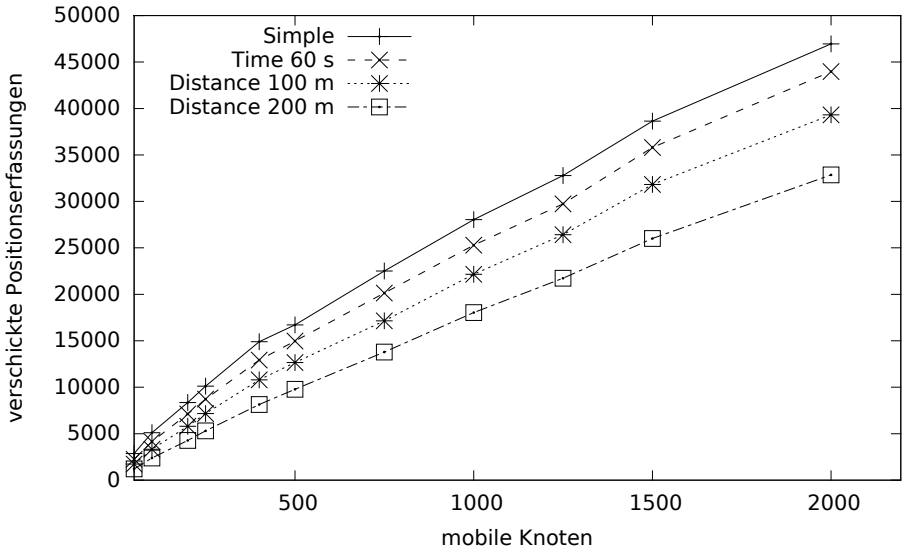
muss, dadurch werden mehr Positionserfassungen unternommen was wiederum zu einer erhöhten Anzahl an versendeten Positionsinformationen führt.

Vergleich verteilte virtuelle Sensoren bei adaptiver Verteilung

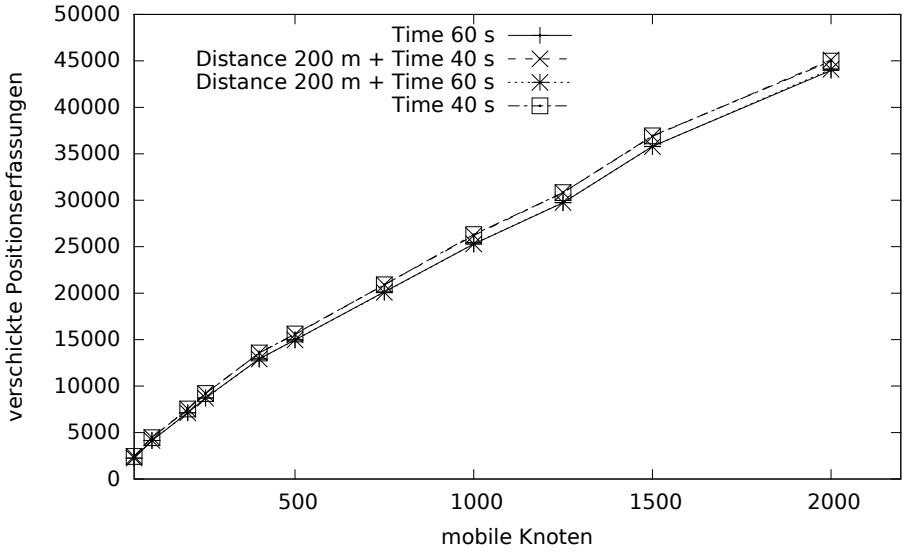
Nachdem im ersten Schritt nur die Anzahl der versendeten Positionserfassungen betrachtet wurde, soll nun die Anzahl der verteilten virtuellen Sensoren, ein weiteres Minimierungskriterium, untersucht werden, dabei wird wieder die Strategie der kartenbasierten adaptiven Verteilung der virtuellen Sensoren angewandt. Die Grafik 6.10a zeigt dazu den Vergleich, wobei das Ergebnis dabei invers zum Ergebnis der versendeten Positionserfassungen ausfällt. Dies ist soweit auch zu erwarten, da mit einer größeren Ungenauigkeit des Servers eine höhere Anzahl der versendeten virtuellen Sensoren einhergeht. Die Grafik 6.10b zeigt eine Detailansicht der vorangegangenen Grafik mit verschiedenen Parametern des Time-based Update-Protokolls, wobei hier deutlich wird, dass das Simple Protokoll die kleinste Anzahl an verteilten virtuellen Sensoren verursacht, wobei der Vorteil gegenüber den Time-based Protokollen nur marginal ist.

Vergleich Gesamtenergie mit adaptiver Verteilung

Vorgreifend auf das nächste Kapitel soll hier nun schon ein Gesamtvergleich der verschiedenen Update-Protokolle gezogen werden. Da für alle Update-Protokolle sich die Strategie der kartenbasierten adaptiven Verteilung als bessere Alternative gegenüber der adaptiven Verteilung mit euklidischen Distanzen oder gar der naiven Verteilung erwiesen hat, basiert die folgende Betrachtung des Gesamtenergieaufwands auf der kartenbasierten adaptiven Verteilung. In Grafik 6.11 ist das Ergebnis dieser Analyse zu sehen, aus Gründen der Lesbarkeit der Ergebnisse ist dieses in Energieaufwand pro Knoten aufgetragen. Mit dieser Analyse lässt sich nun feststellen, dass das Distance-based Update-Protokoll mit einer Abweichung von 100 m den geringsten Energieaufwand für das Gesamtsystem benötigt. Bemerkenswert ist hierbei auch, dass das Simple Protokoll einen geringeren Energieaufwand aufweist als das Distance-based Protokoll mit einer Abweichung von 200 m. Der Grund dafür liegt darin, dass durch die Verwendung des Distance-based Protokolls mit einer solch großen Abweichung die Positionsunsicherheit des Servers relativ groß wird und dem mobilen Knoten im Vergleich zum Simple Protokoll mehr virtuelle Sensoren zugestellt werden. Dadurch muss der mobile Knoten zudem potentiell mehr adaptive Positionserfassungen durchführen, da dem mobilen Knoten nun mehr virtuelle Sensoren bekannt sind, die potentiell in der Nähe liegen können. Die Optimierung der adaptiven Verteilung bei Verwendung des Distance-based Protokolls kann in diesem Szenario zudem nicht angewandt werden, da die adaptive Positionserfassung aktiv ist und diese keine Garantie auf Erkennung der Überschreitung der geforderten Maximaldistanz erfüllen kann. Dies alles führt im Endeffekt dazu, dass die Anwendung des Simple Protokolls zu einem geringeren Energieaufwand führt, als die Anwendung des Distance-based Protokolls mit einer Maximalabweichung von 200 m.

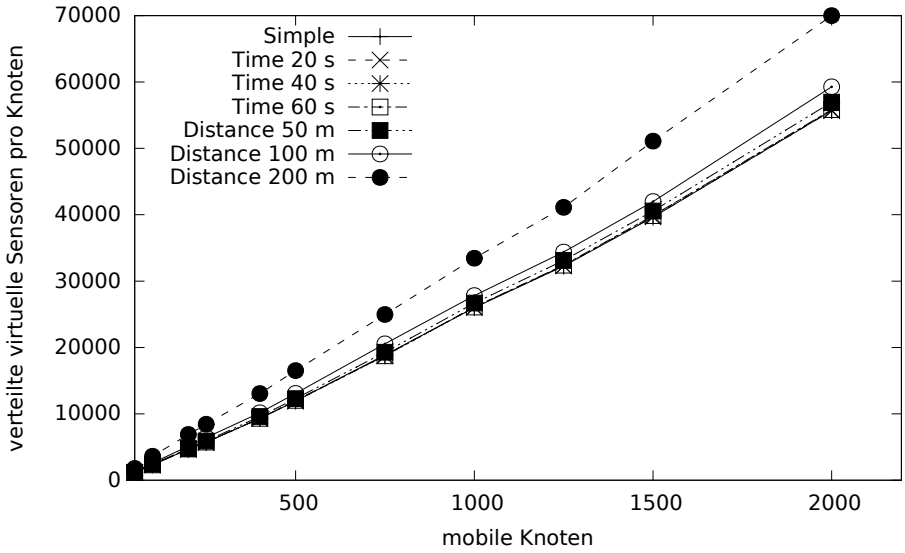


(a) versendete Positionsinformationen mit verschiedenen Update-Protokollen

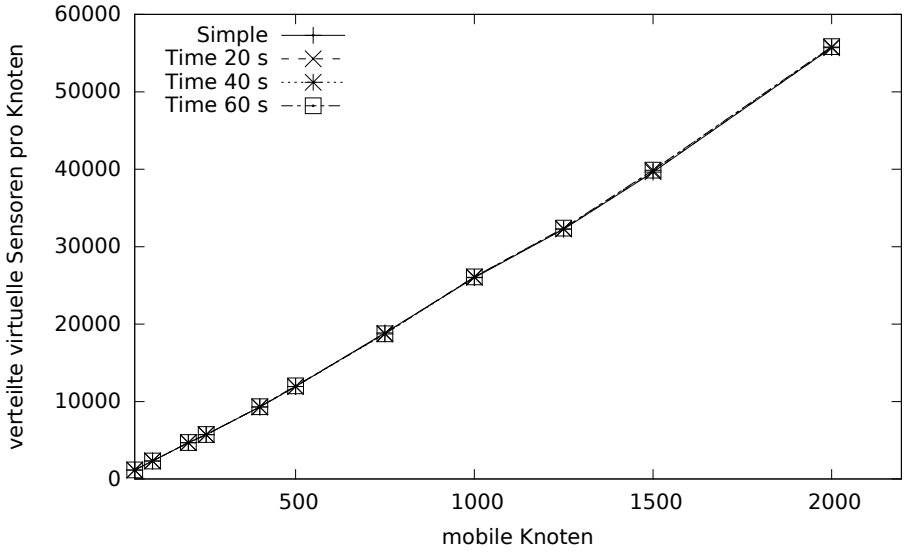


(b) versendete Positionsinformationen mit kombiniertem Update-Protokoll

Abbildung 6.9: Vergleich der Anzahl versendeter Positionsinformationen an den Server mit adaptiver Positionserfassung und kartenbasierter adaptiver Verteilung der virtuellen Sensoren



(a) Anzahl der versendeten virtuelle Sensoren mit verschiedenen Update-Protokollen



(b) Detailansicht der Grafik 6.10a

Abbildung 6.10: Vergleich der Anzahl versendeter virtueller Sensoren mit adaptiver Positionserfassung und kartenbasierter adaptiver Verteilung der virtuellen Sensoren

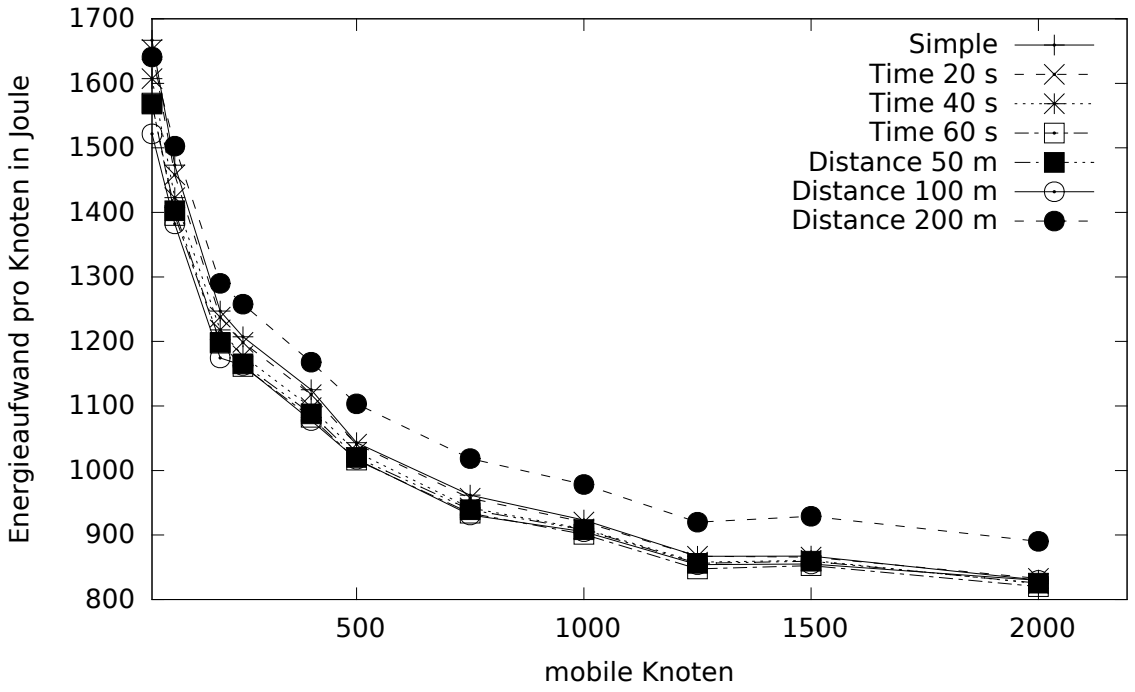


Abbildung 6.11: Gesamtenergieaufwand pro Knoten mit kartenbasierter adaptiver Verteilung der virtuellen Sensoren und adaptiver Positionserfassung

6.2.5 Energievergleich

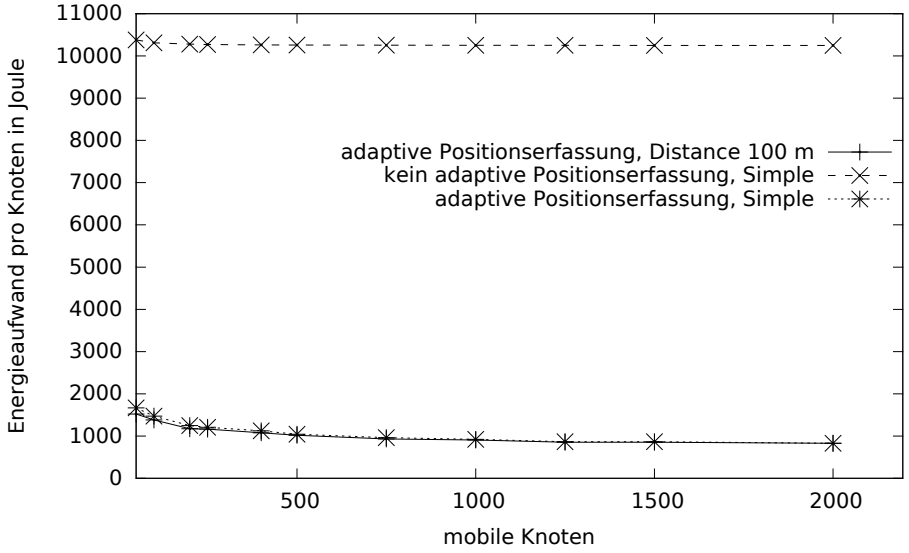
Nachdem sich die vorangegangenen Kapitel sich mit den Auswirkungen verschiedener Parameter auf die Anzahl der versendeten virtuellen Sensoren, die Anzahl der Positionserfassungen und der Anzahl der an den Server versendeten Positionserfassungen beschäftigt hatten, soll nun ein Vergleich des Energieaufwands vorgenommen werden. Dabei werden verschiedene Strategien nach ihrem Gesamtenergieaufwand verglichen welcher sich aus der Addition des Energieaufwands für das Kommunikationssystem und das Positionserfassungssystems ergibt.

Gesamtenergie

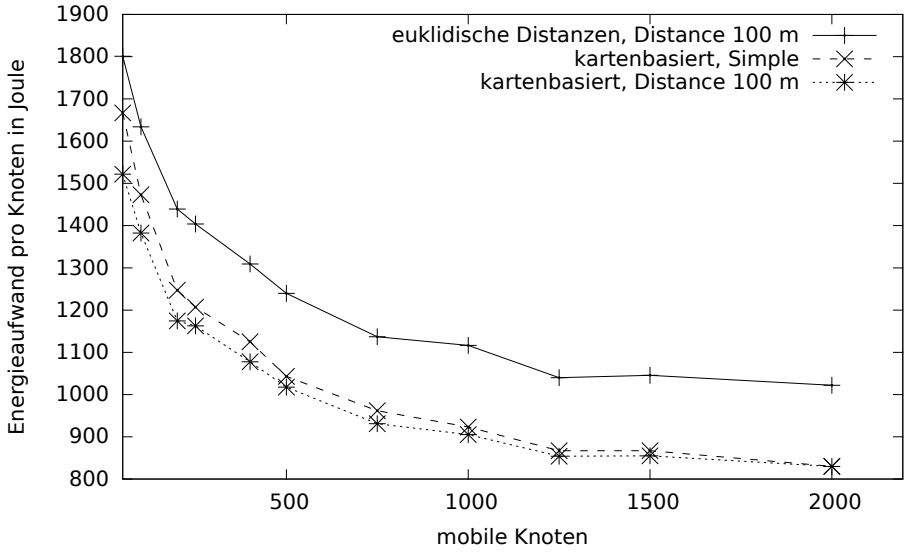
Im folgenden Kapitel soll nun der Gesamtenergieaufwand verglichen werden, das dabei entstehende Ergebnis lässt sich auch als Gesamtergebnis dieser Arbeit deuten, da mit der Betrachtung des Gesamtenergieaufwands die wohl direkteste Vergleichbarkeit der verschiedenen Ansätze mit einer allumfassenden Metrik erreicht wird. Dies ist zudem von Bedeutung, da die Strategie mit dem geringsten Gesamtenergieaufwand auch für reale Anwendungen einen Einfluss hat: Wendet das Gesamtsystem weniger Energie auf, so sinkt auch für einzelne teilnehmende mobile Knoten der Energieaufwand, was für das Gesamtsystem von Vorteil ist, da nun das verwendete Gerät eine längere Betriebszeit besitzt und damit dem Gesamtsystem

länger zur Verfügung stehen kann. Nicht zuletzt hat auch der Besitzer des Geräts davon einen Vorteil, da er nun sein Gerät länger benutzen und trotzdem Daten für das Public Sensing-System liefern kann.
Die Grafik 6.12a zeigt nun dazu den Vergleich verschiedener Strategien, wobei hier nur diejenigen Strategien aufgenommen wurden, die jeweils in den Einzelkriterien wie der Anzahl der verteilten virtuellen Sensoren oder der Anzahl der Positionserfassungen sich als die beste Strategie herausgestellt hatten, zusätzlich wird noch das vom Energieaufwand gesehen beste Update-Protokoll in die Analyse mit aufgenommen. Die Grafik 6.12b zeigt den Energieverbrauch pro teilnehmenden Knoten und zeigt dabei zusätzlich den Einfluss der kartenbasierten Verteilung der virtuellen Sensoren für den Gesamtenergieaufwand auf. Dies zeigt, dass die kartenbasierte Verteilung zusammen mit der adaptiven Positionserfassung und einem geeigneten Update-Protokoll (in diesem Fall das Distance-based Protokoll mit 100 m Abweichung) die für das Gesamtsystem aufgewandte Energie minimiert, wobei dies auch als Fazit der Evaluation gelten kann.

Die vorgestellte Analyse zeigt auch den positiven Effekt auf den Energieaufwand für einzelne Teilnehmer bei steigender Gesamtteilnehmerzahl: Je mehr mobile Knoten an dem Public Sensing-System teilnehmen, desto geringer wird der Energieaufwand pro Knoten, da nun die verteilten virtuellen Sensoren potentiell früher erfasst werden, aus dem System entfernt werden, woraus eine Minimierung des Energieaufwands für den einzelnen Teilnehmer resultiert.



(a) Gesamtenergieaufwand auf Knotenbasis mit adaptiver Verteilung der virtuellen Sensoren



(b) Gesamtenergieaufwand auf Knotenbasis mit adaptiver Positionserfassung

Abbildung 6.12: Vergleich des Gesamtenergieaufwands mit verschiedenen Strategien, mit adaptiver Verteilung der virtuellen Sensoren

7 Zusammenfassung

Dieses Kapitel gibt einen Ausblick auf mögliche weitere Entwicklungen und fasst die vorgestellte Arbeit zusammen.

7.1 Weiterführende Optimierungsansätze

Im Verlauf der Arbeit haben sich an mehreren Stellen noch weitere Potentiale zur Steigerung der Effizienz von Einzelkomponenten oder des Gesamtsystems ergeben:

- Bei der Verwendung der kartenbasierten adaptiven Verteilung der virtuellen Sensoren wird bei der Berechnung des kürzesten Wegs von einem mobilen Knoten zu dem nächstgelegenen virtuellen Sensor die Bewegungsrichtung des mobilen Knoten nicht mit in die Berechnung einbezogen. Mithilfe der Verwendung dieser Information werden potentiell längere Wege gefunden, die zudem das Bewegungsprofil der mobilen Knoten besser abbilden, wodurch eine geringere Anzahl an virtuellen Sensoren an die mobilen Knoten verteilt wird.
- Ein weiteres Optimierungspotential ergibt sich durch eine veränderte Abschätzung der Geschwindigkeit der mobilen Knoten: In dieser Arbeit wurde diese dabei mit v_{max} pessimistisch abgeschätzt, sowohl für die adaptive Verteilung der virtuellen Sensoren als auch für die adaptive Positionserfassung. Durch die Verwendung einer Abschätzung mit der Durchschnittsgeschwindigkeit v_{avg} kann tendenziell die Effizienz des Gesamtsystems verbessert werden, wobei die Effektivität im Vergleich zum naiven Fall potentiell nicht mehr gehalten werden kann.
- Bei den in Kapitel 4.1.1 vorgestellten Update-Protokollen wurde eines der Protokolle aus [LR01] nicht in das System aufgenommen, das Dead-Reckoning Protokoll: Dieses ist eine Verbesserung des Distance-based Protokolls und verwendet anstelle der Distanz seit der letzten Übertragung eine Abschätzung über den potentiellen Weg des mobilen Knotens und sendet nur dann ein Positionsupdate an den Server, wenn die Abschätzung und die tatsächliche Position zu stark divergieren. Der Einbau dieses Protokolls verspricht eine weitere Effizienzsteigerung im Vergleich zu den bereits vorgestellten Protokollen, allerdings hat die Verwendung dieses Protokolls auch großen Einfluss auf die Arbeitsweise der adaptiven Verteilung der virtuellen Sensoren. Zur Verwendung dieses Protokolls muss daher die adaptive Verteilung auf das Dead-Reckoning Protokoll abgestimmt werden, was eine möglicherweise deutlich erhöhte Komplexität der Berechnung der potentiellen Positionen mit sich bringt, weswegen dieses Protokoll auch nicht in den Entwurf der vorliegenden Arbeit integriert wurde.

- Anstelle der Aufgabenverteilung über einen zentralen Server, der den teilnehmenden mobilen Knoten Aufgaben zustellt, kann die Zustellung der Aufgaben auch unter den Knoten selbst per Ad-Hoc-Verbindungen, beispielsweise per Bluetooth, erfolgen. Der potentielle Vorteil liegt dabei im geringeren Energieaufwand für eine Ad-hoc-Verbindung im Vergleich zu einer 3G-Verbindung [BDR12]. Dabei ist allerdings zu bemerken, dass mit diesem Ansatz potentiell nicht die Effektivität eines naiven Systems erreicht werden kann, da der Austausch der Aufgaben per Ad-hoc-Verbindungen nur über eine geringe Distanz funktioniert. Eine Vorhersage der potentiellen Aufgabenverteilung per Ad-hoc-Verbindung durch den Server ist dabei sehr schwer, was die Nachverfolgung des Aufgabenbestands einzelner mobiler Knoten schwierig macht, wodurch der Server möglicherweise über zu wenig Informationen verfügt, um ein effektives System im Vergleich zum naiven Fall zu gewährleisten.

7.2 Abschluss

In dieser Arbeit wurden Strategien zur effizienten Verteilung von Aufgaben in einem Public Sensing-System vorgestellt. Die oberste Prämisse beim Entwurf der verschiedenen Strategien lag dabei darauf, dass diese mindestens die Effektivität eines naiven Ansatzes besitzen. Zusätzlich zur Aufgabenverteilung wurde auch für die Positionsbestimmung der an einem Public Sensing-System teilnehmenden mobilen Knoten eine verbesserte Strategie vorgestellt. Die verschiedenen Ansätze wurden miteinander und hinsichtlich verschiedener Metriken verglichen, wobei zusätzlich zur reinen Anzahl beispielsweise der verschickten Nachrichten auch eine Betrachtung des Energieaufwands getätigt wurde womit schlussendlich die effizienteste Strategie bezüglich des Energieaufwands zur Aufgabenverteilung und Positionserfassung gefunden wurde.
Die Evaluation diente auch dazu, die potentielle Effizienzsteigerung der in dieser Arbeit entwickelten und implementierten Ansätze zu untersuchen, wobei die vorgestellten Verfahren sich als teils deutlich effizienter zeigten als das grundlegende naive Ansatz. Insbesondere der Ansatz der kartenbasierten adaptiven Verteilung der virtuellen Sensoren und die adaptive Positionserfassung haben in der Evaluation die erwünschte Effizienzsteigerung bestätigt, wobei die Kombination der kartenbasierten Verteilung mit adaptiver Positionserfassung und dem Distance-based Update-Protokoll mit einer maximalen Abweichung von 100 m sich als die vom Energieaufwand her gesehen effizienteste Konfiguration erwiesen hat. Diese Konfiguration besitzt gegenüber dem grundlegenden System die Vorteile der adaptiven Positionserfassung, welche die Anzahl der benötigten Positionserfassungen reduziert, die Vorteile der Verwendung eines Lokations-Update Protokolls, welches die Anzahl der an den Server übertragenen Positionserfassungen reduziert, und den Vorteil der adaptiven Verteilung der virtuellen Sensoren mit Kartenunterstützung, wobei diese die Anzahl der verteilten virtuellen Sensoren reduziert.

Nicht zuletzt wurden weitere Möglichkeiten zur Effizienzsteigerung aufgezeigt, die in späteren Projekten Verwendung finden können.

Literaturverzeichnis

[3gp] Third generation partnership project (3gpp). URL http://www.3gpp.org. (Zitiert auf Seite 12)

[BBV09] N. Balasubramanian, A. Balasubramanian, A. Venkataramani. Energy consumption in mobile phones: a measurement study and implications for network applications. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, IMC '09, S. 280–293. 2009. (Zitiert auf den Seiten 12 und 13)

[BDR12] P. Baier, F. Dürr, K. Rothermel. PSense: Reducing Energy Consumption in Public Sensing Systems. In *Proceedings of the 2012 IEEE 26th International Conference on Advanced Information Networking and Applications*, AINA '12, S. 136–143. 2012. (Zitiert auf den Seiten 17 und 70)

[BEH+06] J. Burke, D. Estrin, M. Hansen, A. Parker, N. Ramanathan, S. Reddy, M. B. Srivastava. Participatory sensing. In *Workshop on World-Sensor-Web (WSW'06): Mobile Device Centric Sensor Networks and Applications*, S. 117–134. 2006. (Zitiert auf Seite 17)

[Bre65] J. Bresenham. Algorithm for Computer Control of a Digital Plotter. *IBM Systems Journal*, 4(1):25–30, 1965. (Zitiert auf Seite 46)

[BWDR11] P. Baier, H. Weinschrott, F. Dürr, K. Rothermel. MapCorrect: Automatic Correction and Validation of Road Maps Using Public Sensing. In *36th Annual IEEE Conference on Local Computer Networks (LCN 2011)*, S. 1–8. IEEE Computer Society, Bonn, Germany, 2011. (Zitiert auf Seite 18)

[CEL+06] A. T. Campbell, S. B. Eisenman, N. D. Lane, E. Miluzzo, R. A. Peterson. People-Centric Urban Sensing. In *WICON'06: Proceedings of the 2nd annual international workshop on Wireless internet*, S. 18. ACM, New York, NY, USA, 2006. (Zitiert auf Seite 17)

[CEL+08] A. T. Campbell, S. B. Eisenman, N. D. Lane, E. Miluzzo, R. A. Peterson, H. Lu, X. Zheng, M. Musolesi, K. Fodor, G.-S. Ahn. The Rise of People-Centric Sensing. *IEEE Internet Computing*, 12(4):12–21, 2008. (Zitiert auf Seite 17)

[CHK08] D. Cuff, M. Hansen, J. Kang. Urban Sensing: Out of the Woods. *Communications of the ACM*, 51(3):24–33, 2008. (Zitiert auf Seite 17)

[CKK+08] C. Cornelius, A. Kapadia, D. Kotz, D. Peebles, M. Shin, N. Triandopoulos. Anonysense: privacy-aware people-centric sensing. In *Proceedings of the 6th international conference on Mobile systems, applications, and services*, MobiSys '08, S. 211–224. 2008. (Zitiert auf Seite 17)

[Coh] D. Cohen. Voxel Traversal along a 3D Line. In *Graphics Gems IV*, S. 366–369. AP Professional. (Zitiert auf Seite 46)

[Dij59] E. W. Dijkstra. A Note on Two Problems in Connexion with Graphs. In *Numerische Mathematik*, Band 1, S. 269–271. Springer-Verlag, 1959. (Zitiert auf Seite 31)

[Flo62] R. W. Floyd. Algorithm 97: Shortest path. *Communications of the ACM*, 5(6):345–, 1962. (Zitiert auf Seite 43)

[Jav12] *JavaGeom: geometry with Java*, 2012. URL http://geom-java.sourceforge.net/. (Zitiert auf Seite 42)

[JGr12] *JGraphT – a free Java graph library*, 2012. URL http://jgrapht.org/. (Zitiert auf Seite 42)

[Kan10] E. Kanjo. NoiseSPY: A Real-Time Mobile Phone Platform for Urban Noise Monitoring and Mapping. *Mobile Networks and Applications Journal*, 15(4):562 – 574, 2010. (Zitiert auf Seite 18)

[KBP+08] E. Kanjo, S. Benford, M. Paxton, A. Chamberlain, D. S. Fraser, D. Woodgate, D. Crellin, A. Woolard. MobGeoSen: Facilitating Personal Geosensor Data Collection and Visualization Using Mobile Phones. *Personal and Ubiquitous Computing*, 12:599–607, 2008. (Zitiert auf Seite 17)

[KLGT09] M. B. Kjærgaard, J. Langdal, T. Godsk, T. Toftkjær. EnTracked: energy-efficient robust position tracking for mobile devices. In *Proceedings of the 7th international conference on Mobile systems, applications, and services*, MobiSys '09, S. 221–234. 2009. (Zitiert auf den Seiten 13, 14 und 16)

[KOK09] A. Keränen, J. Ott, T. Kärkkäinen. The ONE Simulator for DTN Protocol Evaluation. In *Proceedings of the 2nd International Conference on Simulation Tools and Techniques*, SIMUTools '09, S. 55:1–55:10. 2009. (Zitiert auf Seite 37)

[LML+10] N. D. Lane, E. Miluzzo, H. Lu, D. Peebles, T. Choudhury, A. T. Campbell. A survey of mobile phone sensing. *Communications Magazine, IEEE*, 48(9):140–150, 2010. (Zitiert auf Seite 17)

[LR01] A. Leonhardi, K. Rothermel. A Comparison of Protocols for Updating Location Information. *Cluster Computing*, 4(4):355–367, 2001. (Zitiert auf den Seiten 21 und 69)

[LYC04] C.-C. Lee, J.-H. Yeh, J.-C. Chen. Impact of inactivity timer on energy consumption in wcdma and cdma2000. In *Proceedings of the Third Annual Wireless Telecommunication Symposium (WTS)*, S. 15–24. IEEE, 2004. (Zitiert auf Seite 12)

[MLEC07] E. Miluzzo, N. D. Lane, S. B. Eisenman, A. T. Campbell. CenceMe – Injecting Sensing Presence into Social Networking Applications. In G. Kortuem, J. Finney, R. Lea, V. Sundramoorthy, Herausgeber, *Smart Sensing and Context*, Band 4793 von *Lecture Notes in Computer Science*, S. 1–28. Springer Berlin / Heidelberg, 2007. (Zitiert auf Seite 17)

[MSN+09] N. Maisonneuve, M. Stevens, M. E. Niessen, P. Hanappe, L. Steels. Citizen Noise Pollution Monitoring. In *Proceedings of the 10th Annual International Conference on Digital Government Research*, dg.o '09, S. 96–103. Digital Government Society of North America, 2009. (Zitiert auf Seite 17)

[OLFM07] Y. Ouyang, Z. Le, J. Ford, F. Makedon. PrivaSense: providing privacy protection for sensor networks. In *Proceedings of the 5th international conference on Embedded networked sensor systems*, SenSys '07, S. 415–416. 2007. (Zitiert auf Seite 17)

[PDR11] D. Philipp, F. Dürr, K. Rothermel. A Sensor Network Abstraction for Flexible Public Sensing Systems. In *Proceedings of the 8th IEEE International Conference on Mobile Ad-Hoc and Sensor Systems*, MASS '11, S. 460–469. IEEE Computer Society, Washington, DC, USA, 2011. (Zitiert auf Seite 17)

[PKG10] J. Paek, J. Kim, R. Govindan. Energy-efficient rate-adaptive GPS-based positioning for smartphones. In *Proceedings of the 8th international conference on Mobile systems, applications, and services*, MobiSys '10, S. 299–314. 2010. (Zitiert auf den Seiten 14 und 15)

[RB12] J. Rula, F. E. Bustamante. Crowd (soft) control: moving beyond the opportunistic. In *Proceedings of the Twelfth Workshop on Mobile Computing Systems & Applications*, HotMobile '12, S. 3:1–3:6. 2012. (Zitiert auf Seite 17)

[RES10] S. Reddy, D. Estrin, M. Srivastava. Recruitment framework for participatory sensing data collections. In *Proceedings of the 8th international conference on Pervasive Computing*, Pervasive'10, S. 138–155. 2010. (Zitiert auf Seite 17)

[SM08] A. Steed, R. Milton. Using Tracked Mobile Sensors to Make Maps of Environmental Effects. *Personal and Ubiquitous Computing*, 12(4):331–342, 2008. (Zitiert auf den Seiten 17 und 18)

[SP01] S. Slijepcevic, M. Potkonjak. Power efficient organization of wireless sensor networks. In *IEEE International Conference on Communications, 2001. ICC 2001.*, Band 2, S. 472 –476 vol.2. 2001. (Zitiert auf Seite 18)

Alle URLs wurden zuletzt am 11.09.2012 geprüft.

Erklärung

Hiermit versichere ich, diese Arbeit selbständig verfasst und nur die angegebenen Quellen benutzt zu haben.

(Steffen Maaß)