Ein robustes Datenmonitoring-Verfahren für Sensornetzwerke

A Robust Data Monitoring Method for Sensor Networks

Volker Turau, Christoph Weyer, Matthias Witt, Technische Universität Hamburg-Harburg

Zusammenfassung Es wird ein neues Datenmonitoring-Verfahren für drahtlose Sensornetzwerke vorgestellt. Das Verfahren kombiniert geografische Hashtabellen mit Aggregation innerhalb des Netzwerks. Anfragen werden in Gebieten abgearbeitet, deren Sensorknoten ihre Werte in Knoten nahe einer festgelegten Lokation innerhalb des Gebietes sammeln, wo sie leicht von außen abgefragt werden können. Das Verfahren ist robust gegenüber Knotenausfällen und -bewegungen und erzeugt sehr geringen Netzwerkverkehr. Dies wird durch verschiedene Simulationen bestätigt. ►►► **Summary** A new data monitoring method for wireless sensor networks is introduced. The method combines geographic hash tables with in-network aggregation. Queries are processed in regions, whose sensor nodes collect their values in nodes close to a determinated location within the region, where they can easily be requested from outside. The method is robust against node failures and movements and generates very little network traffic. This is confirmed by various simulations.

KEYWORDS C.3 [Special-Purpose and Application-Based Systems], B.8 [Performance and Reliability], H.3.3 [Information Search and Retrieval], sensor networks, data aggregation, data-centric storage

1 Einleitung

Drahtlose Sensornetze eröffnen neue Perspektiven für die Überwachung vielfältiger, auch schwer zugänglicher Umgebungen sowohl im natürlichen als auch im industriellen Umfeld. Dabei ist es wünschenswert, dass die Sensorknoten nicht einzeln abgefragt werden, einerseits wegen des hohen Kommunikationsaufwandes, andererseits, weil die Adressen und momentanen Verfügbarkeiten der Knoten möglicherweise nicht bekannt sind. Die Nachrichten tragen als Ziel keine Knotenadressen, sondern geografische Positionen. Es interessieren nicht die Daten bestimmter Knoten, sondern bestimmter geografischer Gebiete, wobei nicht notwendigerweise bekannt ist, welche und wie viele Knoten sich innerhalb der betreffenden Gebiete befinden. Daher ist es sinnvoll, Anfragen in solchen Regionen auszubreiten, wobei z. B. der Mittelwert der Sensoren berechnet werden soll. Die Knoten im Gebiet sorgen selbständig für die regelmäßige Aktualisierung des zu berechnenden Wertes. Im Sensornetz kann es viele solcher Gebiete geben, die sich auch überlappen dürfen und in gewisser Hinsicht nur-lesbare Speicherplätze darstellen. Die Werte können von beliebig vielen Senken aus leicht abgefragt werden, indem eine Anfrage auf den Wert zum jeweiligen Gebiet gesendet wird.

Der vorliegende Beitrag betrifft die höheren Netzwerkschichten in Sensornetzen und behandelt Datenverbreitung und -aggregation sowie die Verarbeitung von Anfragen. Unter einem Sensornetz wird eine Ansammlung von Knoten verstanden, die über ein oder mehrere Sensoren verfügen (z. B. für Temperatur, Luftdruck usw.) und über eine Kommunikationsschnittstelle drahtlos Nachrichten austauschen können. Die Sendereichweite ist beschränkt und wird als einheitlich angesehen, um unidirektionale Verbindungen zu vermeiden. Die Knoten sind in einer Ebene verteilt und besitzen Informationen über ihre geografische Position, z. B. über GPS oder fest in den Knoten kodiert. Die beschränkten Energieressourcen bedingen eine Reduktion der Kommunikation auf ein Minimum.

Weitere Komplexität bringen Knotenbewegungen und -ausfälle. Die hohe Dynamik macht statische Routingverfahren unbrauchbar. Das vorgestellte Verfahren reagiert durch Replikation und periodisches Sen-

den von Daten- und Kontrollnachrichten flexibel auf Änderungen der Topologie.

2 Stand der Forschung

Beim Monitoring von Sensordaten geht es darum, die Daten einer Menge von Sensorknoten (meist in einem zusammenhängenden Gebiet) an eine oder mehrere Senken zu übermitteln. Um die Knoten zum periodischen Senden der Daten zu bringen, muss zuvor die Anfrage im Netz verbreitet werden. Eine Möglichkeit besteht darin, die Anfrage per *Flooding* auszubreiten; Knoten, die die Anfrage erhalten, prüfen, ob sie zum Zielgebiet gehören, und starten ggf. einen Task, der die Sensordaten periodisch zur Senke schickt. Der Nachteil dieser Methode ist eine übermäßig hohe Anzahl an Paketen, da die Anfrage durch das ganze Netzwerk ausgebreitet wird und die Knoten ihre Pakete unabhängig voneinander zurücksenden. Außerdem werden neu hinzugekommene Knoten nicht berücksichtigt.

Ein Verfahren, das für Datenmonitoring vorgeschlagen wurde, ist *Directed Diffusion* [1]. Die Senken können Interessen an bestimmten Nachrichtentypen anmelden. Das Interesse wird mittels *Flooding* an die Nachbarn weitergeleitet, jedoch nicht notwendigerweise an alle; hier lässt das Verfahren einen gewissen Spielraum zu, z. B. könnte nur an die Nachbarn gesendet werden, die näher am Zielgebiet liegen. Durch das *Flooding* des Interesses wird für jede Quelle ein Baum aufgebaut, über den die Daten periodisch zurück an die Senken geschickt werden; das Intervall wird im Interesse mitgeschickt. Die Struktur des Baumes hängt davon ab, von welchen Knoten das Interesse zuerst erhalten wird. Falls die Bäume einer Quelle zu verschiedenen Senken zusammenführen, können die Daten quasi per *Multicast* gesendet werden. Da das Interesse in Abständen wiederholt gesendet wird, findet ständig ein *Flooding* statt.

Das Verfahren, das im nächsten Abschnitt beschrieben wird, basiert auf datenzentrierter Speicherung. Hierbei werden Daten mit Namen versehen und die Kommunikation verwendet diese Namen an Stelle von Knotenadressen, wobei das Ziel aus der Anfrage abgeleitet wird. Geografische Hashtabellen (GHT) [2] sind eine Form der datenzentrierten Speicherung. Hierbei bezieht sich jede Anfrage auf einen Schlüssel, der (z. B. über eine Hashfunktion) auf eine geografische Position abgebildet wird. Die Daten zu diesem Schlüssel werden im Knoten verwaltet, der dieser Position am nächsten ist. Es sind die beiden Nachrichtentypen

PUT(Schlüssel, Wert)

und

GET(Schlüssel)

definiert: PUT speichert einen Wert zu einem Schlüssel, GET besorgt diesen Wert.

GHT baut auf GPSR (*Greedy Perimeter Stateless Routing*) [3] als Routingprotokoll auf. Bei GPSR verwaltet jeder Knoten eine Nachbarschaftstabelle, in der die in Sendereichweite liegenden Knoten und ihre Positionen eingetragen werden. Die Knoten erfahren voneinander, indem sie periodisch Kontrollnachrichten (so genannte *Beacons*) senden. Jeder Knoten, der ein *Beacon* erhält, nimmt dessen Sender in seine Nachbarschaftstabelle auf. Der Eintrag wird wieder entfernt, wenn über eine festgelegte Zeitspanne kein *Beacon* mehr empfangen wird. Durch das periodische Senden der *Beacons* werden Knotenausfälle und Änderungen der Topologie erkannt.

GPSR arbeitet in zwei verschiedenen Modi: dem Greedy- und dem Perimeter-Modus. Das Senden eines Paketes beginnt im Greedy-Modus, wobei das Paket zu dem Nachbarn geroutet wird, der dem Ziel am nächsten ist. Gibt es keinen Nachbarn, der näher am Ziel ist als der eigene Knoten, wird in den Perimeter-Modus gewechselt, wobei das Paket mittels der Rechte-Hand-Regel so lange entlang von Kanten einer Planarisation des Nachbarschaftsgraphen gesendet wird, bis es zu einem Knoten gelangt, der näher am Ziel liegt.

Der Perimeter-Modus wird bei GHT für die Bildung so genannter *Home*-Knoten ausgenutzt. Da sich an der Position, auf die ein bestimmter Schlüssel abgebildet wird, kein Sensorknoten befindet[1], wechselt GPSR im Knoten, der der Position am nächsten ist, in den Perimeter-Modus und sendet die Nachricht einmal um die Position herum (über den so genannten *Home*-Perimeter), bis sie wieder am Ausgangsknoten ankommt. Im originalen GPSR wird die Nachricht an dieser Stelle verworfen, weil kein Knoten erreichbar ist, der näher am Ziel liegt; bei GHT wird der Knoten jedoch zum *Home*-Knoten für den Schlüssel. Der *Home*-Knoten ist für die Speicherung des Wertes für den Schlüssel verantwortlich; er empfängt PUT-Nachrichten und beantwortet GET-Nachrichten.

Durch Replikation der Daten des *Home*-Knotens in allen Knoten des *Home*-Perimeters wird gewährleistet, dass bei Ausfall des *Home*-Knotens ein anderer Knoten dessen Aufgabe übernimmt: Der *Home*-Knoten schickt periodisch Refresh-Nachrichten um den *Home*-Perimeter; erhält ein Knoten innerhalb der doppelten Zeitspanne keinen Refresh, schickt er einen eigenen Refresh und wird selbst zum *Home*-Knoten, wenn er diesen nach einem Umlauf um den *Home*-Perimeter zurückerhält. Ein Knoten, der einen Refresh empfängt, aber selbst näher an der Position liegt als der Initiator des Refreshs, verwirft den Refresh und schickt einen eigenen.

[1] Der Fall, dass sich exakt an der Position des Schlüssels ein Knoten befindet, ist zu unwahrscheinlich, um in der Realität aufzutreten; selbst wenn er tatsächlich auftreten sollte, führt dies nicht zum Versagen des Verfahrens, da in diesem Fall der *Home*-Perimeter nur aus einem einzigen Knoten besteht.

3 Der Algorithmus

Der vorgeschlagene Algorithmus ist eine Erweiterung von [4] und wertet Sensordaten innerhalb von Gebieten aus. Er unterstützt keine Ad-hoc-Anfragen, sondern Anfragen, die über einen längeren Zeitraum ausgewertet werden. Die Gebiete müssen folgende Eigenschaften besitzen:

(1) Die Zugehörigkeit eines Knotens zum Gebiet muss einfach überprüfbar sein.
(2) Der Nachbarschaftsgraph der Knoten innerhalb des Gebietes muss zusammenhängend sein.

Während der erste Punkt eine einfache Berechnung der Mitgliedschaft der einzelnen Knoten ermöglicht, garantiert der zweite Punkt, dass die Information der Mitgliedschaft alle Knoten des Gebietes erreicht. Die Einhaltung des zweiten Punktes ist allerdings nicht einfach nachzuprüfen. Es genügt i. A., wenn die Knotendichte im Netzwerk hinreichend groß ist.

Alle Knoten innerhalb des Gebietes liefern den Wert, den ihr Sensor misst; der Gesamtwert wird als Funktion über den Einzelwerten berechnet (z. B. Durchschnitt, Maximum oder Median). Eine Anfrage besteht somit aus der Beschreibung des Zielgebietes (s. obiger Punkt 1) sowie der zu berechnenden Funktion und dem zu messenden Wert. Dies muss nicht zwangsläufig direkt ein Sensorwert sein (z. B. Temperatur), sondern kann sich auch aus mehreren Sensorwerten zusammensetzen (z. B. Temperatur aller Knoten, deren Luftdruck über einer bestimmten Schwelle liegt; die anderen Knoten senden dann keine Werte).

Anfragen werden über Schlüssel identifiziert, die global eindeutig sind. Die gesamte Anfrage – bestehend aus Position und Struktur des Zielgebietes, zu messenden Sensordaten und Aggregationsfunktion – könnte den Schlüssel bilden, was jedoch wegen der großen Länge unpraktikabel ist. Angemessen ist beispielsweise die Berechnung des Schlüssels über eine Hashfunktion aus einer zeichenkettenbasierten Form der Anfrage. Diese Hashfunktion muss bei den Senken bekannt sein, braucht jedoch nicht auf den Knoten im Sensornetz vorzuliegen.

Der Algorithmus verwendet sieben Nachrichtentypen: SENDQUERY, REFRESH, FLOODQUERY, COLLECT, GET, SENDVALUE und PUT. Um eine Anfrage im Netz zu verbreiten, genügt es, dass eine Senke eine SENDQUERY-Nachricht mit der Anfrage und dem entsprechenden Schlüssel mittels GPSR zu einem Punkt innerhalb des Zielgebietes sendet, der so genannten Lokation. Es muss eine Vorschrift vorliegen, mittels derer sich zu einer Anfrage eine eindeutige Lokation bestimmen lässt. Die Vorschrift ist Teil der Implementierung des Verfahrens. Das Ziel bei der Wahl der Lokation ist die Minimierung des Kommunikationsaufwandes innerhalb des Gebietes. Möglich sind hier z. B., je nach geometrischer Form des Gebietes, Mittelpunkt oder Schwerpunkt.

Die SENDQUERY-Nachricht kommt wie bei GHT beim *Home*-Knoten an, wobei gleichzeitig der *Home*-Perimeter aufgebaut wird. Wählt man als Lokation den Mittelpunkt des Gebietes, so liegt der *Home*-Knoten nah am Zentrum. Das Refresh-Protokoll funktioniert genauso wie beim originalen GHT-Verfahren und sorgt für Fehlertoleranz, indem auch bei Knotenausfällen und Knotenbewegungen immer ein *Home*-Knoten gefunden wird. Hierzu werden periodisch REFRESH-Nachrichten verschickt.

Der *Home*-Knoten sendet nun ein FLOODQUERY mit der Anfrage und dem entsprechenden Schlüssel an die *Broadcast*-Adresse, woraufhin ein *Restricted Flooding* stattfindet: Jeder Knoten, der die Nachricht erhält, prüft, ob er sich innerhalb des Zielgebietes befindet. Falls dies zutrifft (und der Knoten die Nachricht innerhalb einer kurzen Zeitspanne nicht schon einmal erhalten hat), sendet er sie weiter[2] und startet einen Timer für die Anfrage. Wenn dieser Timer abläuft, prüft der Knoten, ob er noch im Zielgebiet liegt, und sendet im positiven Fall ein COLLECT an die Lokation und startet den Timer erneut. In die COLLECT-Nachricht schreibt der Knoten den Schlüssel, seine Adresse und den von ihm gemessenen Wert. Knoten, die auf dem Weg zum *Home*-Knoten besucht werden, fügen ihre Adresse und ihren Wert hinzu und setzen ihren Timer zurück. Auf diese Weise wird erreicht, dass diese Knoten in Zukunft keine eigenen COLLECT-Nachrichten mehr initialisieren, da sie vor Ablauf des Timers eine COLLECT-Nachricht erhalten, an die sie ihre Daten anhängen. Nur die Knoten am Rand des Gebietes initialisieren regelmäßig eigene COLLECT-Nachrichten, da sie keine solchen empfangen. Der *Home*-Knoten sammelt alle COLLECT-Nachrichten und verwaltet für jeden Schlüssel eine Tabelle, in der die Knoten, ihre Werte und die Zeitpunkte der letzten Messungen eingetragen werden. Liegt die letzte Messung eines Knotens zu lange zurück (z. B. weil er ausgefallen ist oder sich aus dem Zielgebiet herausbewegt hat), so wird sein Eintrag aus der Tabelle entfernt.

Die Anzahl der Nachrichten kann noch weiter verringert werden, indem Knoten, deren Timer ablaufen, nur dann ein COLLECT senden, wenn sich ihr Wert seit der letzten Messung geändert hat oder eine festgesetzte Zeitspanne abgelaufen ist, in der in jedem Fall gesendet wird.

Der *Home*-Knoten sendet erneut ein FLOODQUERY, wenn die Anzahl der Knoten, von denen neue Werte vorliegen, unter eine festgelegte Schwelle sinkt oder ein längerer Timer abläuft. So erhalten auch Knoten die Anfrage, die sich neu in

[2] Bei geringer Knotendichte könnte es vorkommen, dass bei dieser Verfahrensweise einige Knoten innerhalb des Zielgebietes nicht erreicht werden. Die zweite geforderte Eigenschaft verhindert dies.

das Gebiet hineinbewegt haben oder temporär nicht verfügbar waren.

Soll der Wert der Anfrage von einer Senke aus abgefragt werden, so sendet diese ein GET mit dem entsprechenden Schlüssel an die Lokation. Wenn der *Home*-Knoten diese Nachricht erhält, berechnet er mit Hilfe seiner Tabelle den Wert aus allen Einzelwerten und sendet diesen in einer SENDVALUE-Nachricht zurück. Somit findet das Abfragen des Wertes mit äußerst geringem Kommunikationsaufwand statt.

Der vorgestellte Ansatz hat den Nachteil, dass der *Home*-Knoten sehr schnell zum *Hot Spot* wird, weil alle Knoten innerhalb des Zielgebietes an ihn senden. Um dies zu umgehen, kann eine zweistufige Hierarchie verwendet werden, indem das Zielgebiet in mehrere Teilgebiete partitioniert wird. Für die Partitionierung gibt es verschiedene Möglichkeiten:

- eine einfache geometrische Unterteilung in gleich große Gebiete
- eine Unterteilung unter Berücksichtigung der Netztopologie (z. B. Knotendichte, Umgehung von Kommunikationsengpässen usw.)
- Ausnutzung eines Clustering-Schemas, z. B. über die Konstruktion eines Spannbaums [5]

Jeder Knoten sendet die COLLECT-Nachrichten an die Lokation seines Teilgebietes. Damit bekommt jedes Teilgebiet seinen eigenen *Home*-Knoten, der die Werte sammelt. Falls es sich bei der zu berechnenden Funktion um eine zerlegbare Funktion [6] handelt, können die *Home*-Knoten der Teilgebiete bereits den Wert der Anfrage für ihr Gebiet berechnen. Zerlegbare Funktionen sind z. B. Minimum, Maximum und Durchschnitt, nicht jedoch der Median. Die *Home*-Knoten der Teilgebiete senden periodisch ein PUT an den *Home*-Knoten des gesamten Gebietes. PUT kann zusätzlich zu Schlüssel und Wert noch weitere Daten enthalten, die zur Berechnung des Wertes benötigt werden. Für die Berechnung des Durchschnitts muss z. B. die Summe der Werte s und die Anzahl der Knoten c mitgeliefert werden. Der endgültige Wert berechnet sich dann durch $\frac{s}{c}$. Der *Home*-Knoten des gesamten Gebietes braucht nun nicht mehr die Adressen aller Knoten in seiner Tabelle zu verwalten, sondern speichert lediglich die Lokationen der Teilgebiete, deren Daten sowie den Zeitpunkt der letzten PUT-Nachricht.

Falls die eingangs festgelegten Randbedingungen in der Realität nicht exakt gegeben sind, so funktioniert das Verfahren dennoch, liefert allerdings nicht notwendigerweise korrekte Werte. Wenn einige Knoten außerhalb des Gebietes sich für zugehörig halten oder einige Knoten innerhalb für nicht zugehörig, liefern sie fälschlich ihre Werte mit bzw. nicht mit. Ist der Nachbarschaftsgraph nicht zusammenhängend, fehlen die Werte einiger Knoten. Durch eine ausreichend hohe Knotendichte wird dies allerdings vermieden.

4 Simulationsergebnisse

Um das beschriebene Verfahren zu evaluieren, wurde es in den Netzwerksimulator *ns*-2 [7] implementiert. Durch die detaillierte Simulation der physikalischen Umgebung können realistische Ergebnisse erzielt werden. Als MAC-Schicht kam IEEE 802.11 zum Einsatz. Die Knoten senden zufällige Sensordaten, die sich bei jeder Abfrage mit 30%iger Wahrscheinlichkeit gleich verteilt um maximal $\pm 0{,}5$ ändern. Als Zielgebiete werden Rechtecke angegeben, wobei der Mittelpunkt die Lokation bildet, an die gesendet wird. Die Aufteilung in Teilgebiete findet statt, indem jeweils in X- und Y-Richtung so lange halbiert wird, bis die Kantenlänge ein festgelegtes Maximum nicht überschreitet. Die wichtigsten Simulationsparameter sind in Tabelle 1 zusammengefasst.

Der erste Simulationslauf vergleicht das vorgestellte Verfahren mit *Directed Diffusion*. Hierfür wird auf die in der Distribution von *ns*-2 enthaltene Implementierung von *Diffusion* zurückgegriffen. Es werden 300 Sekunden simuliert; die Topologie besteht aus 500 zufällig angeordneten Knoten, wobei die 1 bis 15 Senken sich am rechten Rand der Topologie befinden, das 120×120 m große Zielquadrat in der Mitte. Die SENDQUERY-Nachricht wird nach 20 s verschickt; die periodischen GET-Anfragen starten bei 25 s. Bei *Diffusion* senden die Senken nach 20 s das Interesse aus; die Quellen innerhalb des Zielquadrates, die im Simulationsskript manuell ausgewählt wurden, beginnen bei 25 s mit dem Senden von Daten.

Bild 1 zeigt das Ergebnis der Simulation. Beim hier vorgestellten Verfahren senden die Senken dabei einmal verschiedene Schlüssel und das andere Mal dieselben. Somit senden im einen Fall alle Senken verschiedene Anfragen und im anderen Fall die gleichen. Alle Anfragen beziehen sich auf dasselbe Zielgebiet, allerdings muss bei unterschiedlichen Anfragen jeder Knoten pro Anfrage ein separates COLLECT bzw. PUT senden. Obwohl jeder Knoten einmal pro Sekunde ein *Beacon* sendet, werden dennoch weitaus weniger Pakete als bei *Diffusion* verschickt, da dort durch das ständig neue Ausbreiten des Inter-

Tabelle 1 Simulationsparameter.

Knotendichte	1 Knoten / 245 m^2
Sendereichweite	40 m
Beacon-Intervall (GPSR)	1 s
Perimeter-Refresh-Intervall	10 s
Anfrage-Intervall	5 s
COLLECT-/PUT-Intervall	5 s
Refresh der Anfrage	30 s

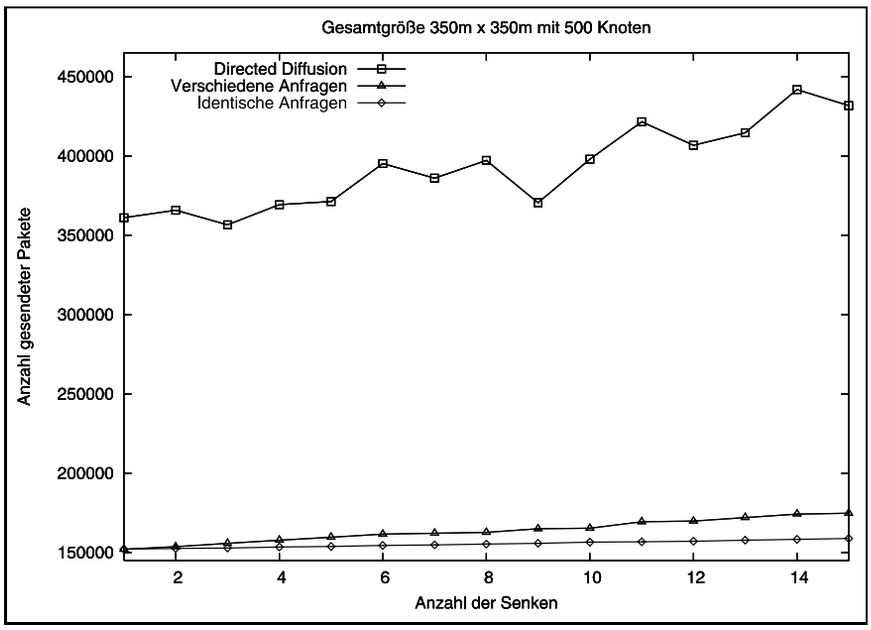

Bild 1 Vergleich mit *Directed Diffusion*.

esses deutlich mehr *Flooding* stattfindet, welches darüber hinaus das gesamte Netz erfasst, während es beim *Restricted Flooding* auf das Zielgebiet beschränkt ist. Auch der Anstieg bei mehreren Senken ist beim neuen Verfahren geringer, da eine Abfrage von einer Senke aus fast nichts kostet.

Während das vorige Experiment noch ohne Unterteilung des Zielgebietes ablief, wird bei den folgenden Experimenten die Auswirkung einer Unterteilung untersucht. Bild 2 zeigt die Anzahl der gesendeten Pakete in Abhängigkeit von der Größe des Zielgebietes. Die Kantenlänge wird von 60 m schrittweise bis auf 320 m erhöht, womit das Zielgebiet schließlich fast die gesamte Topologie von 350 × 350 m umfasst. Es findet einmal keine Unterteilung statt, dann eine Unterteilung in vier und schließlich in 16 Teilquadrate. Während bei kleinen Zielgebieten die meisten Pakete bei 16-facher Unterteilung verschickt werden und keine Unterteilung die beste Wahl ist, kehrt sich das Verhältnis mit zunehmender Größe der Zielregion um, wobei im Fall ohne Unterteilung der stärkste Anstieg zu verzeichnen ist. Wenn die Teilgebiete zu wenig Knoten enthalten, ist der Overhead durch die vielen *Home*-Perimeter und das Senden der Werte über den Umweg der lokalen *Home*-Knoten zu groß. Bei größeren Zielgebieten lohnt sich eine Unterteilung, weil die weit von der Lokation entfernten Knoten ihre Werte auf kurzem Weg zu den lokalen *Home*-Knoten senden können.

Um die Auswirkungen der Unterteilung auf *Hot Spots* zu untersuchen, wurden die Ergebnisse aus den Simulationsläufen von 20 unterschiedlichen zufallsgenerierten Topologien mit je 1020 Knoten gemittelt. Es wurden jeweils 50 Sekunden simuliert. Bild 3 zeigt die Ergebnisse, wobei jeweils die maximale Anzahl der empfangenen Pakete pro Knoten angegeben ist (Streuung und Mittelwert), allerdings ohne Berücksichtigung der *Beacons*. Im Fall ohne Unterteilung ist es der *Home*-Knoten, dessen Belastung mit zunehmender Größe des Zielgebietes stark anwächst. Bei 16-facher Unterteilung ist die maximale Belastung am geringsten, jedoch noch nicht bei sehr kleinen Zielgebieten, wo die Anzahl der Knoten pro Teilgebiet noch zu gering ist. Bei den großen Zielgebieten, die nahe am Rand liegen, kann die maximale Belastung u. U. ebenfalls größer sein als bei vierfacher Unterteilung, weil der *Home*-Perimeter um die gesamte Netzwerkgrenze läuft, wenn der *Home*-Knoten am Rand des Netzes liegt. Dies ist eine bekannte Schwachstelle von GPSR und kann dadurch umgangen werden, dass man auf eine Unterteilung verzichtet, wenn das Zielgebiet am Rand des Netzes liegt.

Der Vergleich der Ergebnisse mit und ohne *Beacons* zeigt auch, dass der Anteil der *Beacons* über

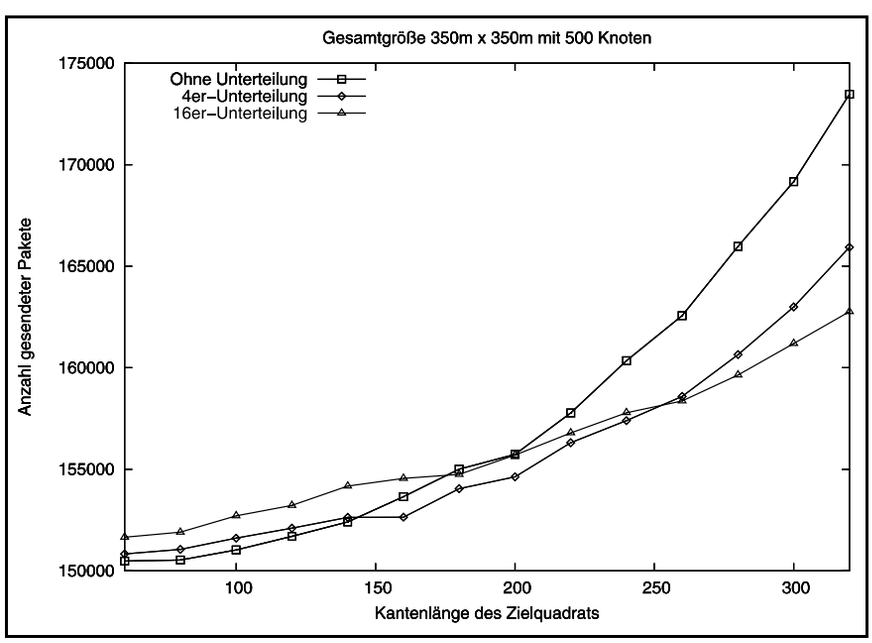

Bild 2 Unterteilung des Zielgebietes.

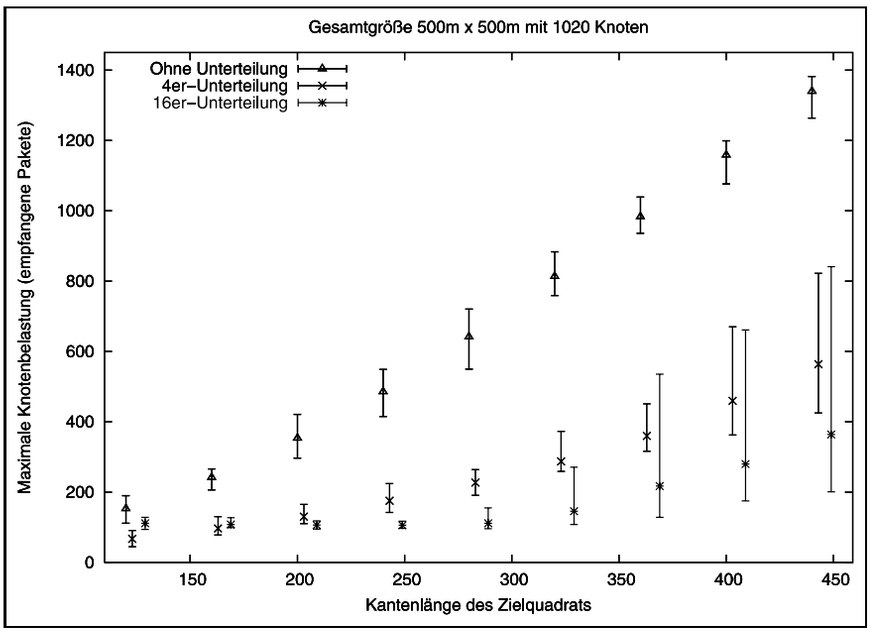

Bild 3 *Hot Spots* bei Unterteilung.

90% der gesendeten Pakete ausmacht. Die Änderung des *Beacon*-Intervalls hat also signifikante Auswirkungen auf den Kommunikationsaufwand. Das Intervall in den Simulationen war mit 1s recht kurz; eine Verlängerung führt dazu, dass der Netzwerkverkehr noch deutlich geringer wird, als er ohnehin schon ist.

5 Fazit

In diesem Beitrag wurde ein neues Datenmonitoring-Verfahren für Sensornetzwerke vorgestellt, das Anfragen in geografischen Gebieten bearbeitet und die Daten selbständig aggregiert. Durch Replikation der Daten des *Home*-Knotens in den Knoten des *Home*-Perimeters und durch regelmäßige Auffrischung der Anfrage ist das Verfahren sehr robust gegenüber Knotenausfällen und Knotenbewegungen. Außerdem wird durch die hierarchische Unterteilung des Zielgebietes die Kommunikationslast besser auf die Knoten verteilt, was zu einem gleichmäßigeren Energieverbrauch führt.

Zukünftige Untersuchungen sollen den Kommunikationsaufwand noch weiter einschränken. Der meiste Netzwerkverkehr findet beim *Restricted Flooding* und durch das Senden der *Beacons* statt. Für das *Restricted Flooding* könnten probabilistische Verfahren eingesetzt werden, wobei jeder Knoten die Anfrage nur mit einer bestimmten Wahrscheinlichkeit weiterleitet. Dadurch wird es allerdings schwer zu garantieren, dass alle Knoten des Zielgebietes die Anfrage erhalten.

Auf die *Beacons*, die über 90% der Kommunikation ausmachen, kann man verzichten, indem statt GPSR ein geografisches Routing eingesetzt wird, das ohne *Beacons* auskommt und die Nachrichten „auf gut Glück" in die richtige Richtung schickt.

Außerdem ist noch zu untersuchen, wie eine Anfrage wieder aus dem Netz entfernt werden kann. Bislang bleibt eine Anfrage, einmal ausgebreitet, im Netz und liegt in mehreren Knoten vor, die sich evtl. gar nicht mehr im *Home*-Perimeter befinden und ggf. später wieder zum *Home*-Knoten werden, wenn der *Home*-Perimeter bereits abgebaut wurde.

Literatur

[1] *C. Intanagonwiwat, R. Govindan, and D. Estrin*. Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks. In: Proc. Mobicom 2000, pp. 56–67.

[2] *S. Ratnasamy, B. Karp, S. Shenker, D. Estrin, R. Govindan, L. Yin, and F. Yu*. Data-Centric Storage in Sensornets with GHT, A Geographic Hash Table. In: Mobile Networks and Applications (2003) Vol. 8(4), pp. 427–442.

[3] *B. Karp and H.T. Kung*. GPSR: Greedy Perimeter Stateless Routing for Wireless Networks. In: Proc. Mobicom 2000, pp. 243–254.

[4] *V. Turau and C. Weyer*. Location-aware In-Network Monitoring in Wireless Sensor Networks. In: Band der GI Jahrestagung (2) 2004, S. 355–359.

[5] *S. Banerjee and S. Khuller*. A Clustering Scheme for Hierarchical Control in Multi-hop Wireless Networks. In: Proc. IEEE Infocom 2001, pp. 1028–1037.

[6] *J. Zhao, R. Govindan, and D. Estrin*. Computing Aggregates for Monitoring Wireless Sensor Networks. In: Proc. First IEEE Int'l Workshop on Sensor Network Protocols and Applications (2003), pp. 139–148.

[7] The Network Simulator – *ns*-2. http://www.isi.edu/nsnam/ns/.


1


2


3

1 Prof. Dr. Volker Turau ist seit 2002 Professor im Arbeitsbereich Telematik an der Technischen Universität Hamburg-Harburg. Seine Forschungsinteressen umfassen verteilte Informationssysteme, Sensornetzwerke, E-Learning-Umgebungen sowie die Integration unternehmensweiter Anwendungen. Von 1977 bis 1983 studierte und promovierte er an der Johannes-Gutenberg-Universität in Mainz. Anschließend hatte er eine Postdoktorandenstelle an der Universität Manchester in England und war als wissenschaftlicher

Mitarbeiter an der Universität Karlsruhe tätig. Volker Turau ist seit Jahren aktives Mitglied in zahlreichen Programm-Komitees. Er hat drei Bücher über Web-Technologien für den unternehmensweiten Einsatz und ein Buch zur algorithmischen Graphentheorie geschrieben.
Adresse: Technische Universität Hamburg-Harburg, Schwarzenbergstraße 95, 21073 Hamburg, Tel.: +49-40-42878-3530, Fax: +49-40-42878-2581, E-Mail: turau@tuhh.de

2 Dipl.-Inform. (FH) Christoph Weyer studierte bis 1995 Informatik an der FH Wiesbaden. Nach seinem Studium arbeitete er als selbständiger IT-Berater und in einigen Drittmittel-Projekten an der FH Wiesbaden, vor allem im Bereich des Managements von verteilten Anwendungen. Seit 2003 ist er im Arbeitsbereich Telematik der Technischen Universität Hamburg-Harburg angestellt. Seine Forschungsschwerpunkte liegen vor allem im Bereich von Sensornetzen und der vertikalen Integration in der Automatisierung.
Adresse: Technische Universität Hamburg-Harburg, Schwarzenbergstraße 95, 21073 Hamburg, Tel.: +49-40-42878-3375, Fax: +49-40-42878-2581, E-Mail: c.weyer@tuhh.de

3 Dipl.-Inform. Matthias Witt hat Informatik an der Universität Hamburg studiert und sein Studium 2003 mit dem Diplom abgeschlossen. Seit 2004 ist er wissenschaftlicher Mitarbeiter am Arbeitsbereich Telematik der Technischen Universität Hamburg-Harburg und promoviert über Sensornetze.
Adresse: Technische Universität Hamburg-Harburg, Schwarzenbergstraße 95, 21073 Hamburg, Tel.: +49-40-42878-3448, Fax: +49-40-42878-2581, E-Mail: matthias.witt@tuhh.de