Datenverbreitung und Aggregation, Routing

ETH Zürich, Institut für Pervasive Computing Prof. F. Mattern
Seminar Verteilte Systeme, Thema „Sensornetze“, SS2003
Student: Pascal von Rickenbach
Betreuer: Thomas Schoch

Abstract

Die Art wie Daten in einem Sensornetz verbreitet werden, ist eine wichtige Designentscheidung. Denn erst dank der Datenverbreitung sind die einzelnen Sensorknoten in der Lage, sich untereinander so zu koordinieren, dass sie die Aufgaben eines Sensornetzes erfüllen können.
Ein Sensornetz besteht dabei aus einer grossen Anzahl von Knoten, wobei diese nur über eingeschränkte Hardware und limitierte Energieressourcen verfügen. Der Bericht führt zunächst aus, wieso sich bestehende Routing-Protokolle für Ad-hoc-Netze nicht für den Einsatz in Sensornetzen eignen, und statt dessen speziell für Sensornetze konzipierte Protokolle für die Datenverbreitung eingesetzt werden sollten.
Anschliessend werden vier existierende Protokolle vorgestellt, welche die Datenverbreitung in Sensornetzen unterstützen. Es handelt sich dabei um die Protokolle SPIN, Directed Diffusion, LEACH und TTDD, die jeweils unterschiedliche Ansätze verfolgen. So wird mit SPIN eine Variante des klassischen Flooding, mit Directed Diffusion ein gradientenbasiertes Verfahren und mit LEACH eine Clustering-Variante vorgestellt.

Einführung

Der Datenverbreitung in Sensornetzen kommt eine grosse Bedeutung zu, da ein solcher Verbund aus vielen Sensorknoten erst mit ihrer Hilfe seine Aufgaben erfüllen kann. So müssen Daten nicht nur zur Koordination der einzelnen Knoten untereinander verbreitet werden, sondern auch, um von aussen kommende Anfragen zu verarbeiten, und die dadurch erzeugten Resultate zurückzuliefern.

Sensornetze können auch als eine Unterklasse von drahtlosen Ad-hoc-Netzen betrachtet werden, da sie unter anderem beide ihre Daten mittels Multi-Hop-Kommunikation von einem Sender zu einem Empfänger routen. Für Ad-hoc-Netze sind dazu bereits viele verschiedene Routing-Protokolle bekannt. Weil aber die Art der Anwendungen und die Anforderungen an das Routing in Sensornetzen und Ad-hoc-Netzen verschieden sind, sollten keine Ad-hoc-Routing-Protokolle in Sensornetzen verwendet werden. Im folgenden werden die Gründe, die gegen den Einsatz bereits existierende Protokolle sprechen, im Detail erläutert.

In Sensornetzen kommt meistens ein anderes Kommunikationsparadigma als in Ad-hoc-Netzen zum Einsatz. Ad-hoc-Routing-Protokolle sind adressbasiert, wohingegen in Sensornetzen hauptsächlich von datenbasierter Kommunikation gesprochen wird. In Ad-hoc-Netzen werden Knoten anhand einer Adresse eindeutig identifiziert. Bei der Datenverbreitung werden die Nachrichten dann mittels „point-to-point“-Kommunikation vom Sender zum Empfänger geroutet. Dies ist für Ad-hoc-Netze auch passend, da die Routing-Protokolle eine Plattform für ganz verschiedene Anwendungen zur Verfügung stellen müssen. Im Unterschied dazu werden in Sensornetzen die Daten aufgrund ihres Inhalts geroutet. Die Empfänger einer Nachricht werden dabei in der Nachricht selbst mit Hilfe von Attributen spezifiziert. Anders als bei der direkten Adressierung des Empfängers, weiss ein Sender also im allgemeinen nicht, welche Knoten die Nachricht empfangen. Dadurch wird der Unzuverlässigkeit der einzelnen Sensorknoten Rechnung getragen. Diese Art der Kommunikation impliziert aber auch, dass eine Nachricht meistens nicht nur an einen Empfänger gerichtet ist, sondern an mehrere. Man spricht in Sensornetzen also eher von einer „many-to-one“- bzw. „one-to-many“-Kommunikation als von einer „point-to-point“-Kommunikation, wie sie in Ad-hoc-Netzen verwendet wird. Weiter ist es möglich, dass die Protokolle zur Datenverbreitung in Sensornetzen „application-aware“ sind, weil der Anwendungszweck des Netzes bereits vor der Inbetriebnahme festgelegt ist. Die Protokolle können also im voraus für die jeweilige Anwendung optimiert werden.

Der zweite entscheidende Faktor, der gegen die Verwendung von Ad-hoc-Routing-Protokollen spricht, ist die Tatsache, dass in Sensornetzen die Energieressourcen limitiert sind, und darum der Energieeffizienz von Kommunikationsprotokollen grösste Bedeutung zukommt. Ad-hoc-Routing-Protokolle sind aber in erster Linie auf einen hohen Durchsatz, eine kleine Latenz und eine hohe Knotenmobilität optimiert, und nicht dafür, möglichst wenig Energie zu verbrauchen. In Sensornetzen sind diese drei Punkte aber nur von sekundärer Bedeutung, da die Protokolle in erster Linie auf die Energieeffizienz hin optimiert sein sollten. Ein Sensornetz soll schliesslich seine Aufgaben so lange wie möglich erfüllen können, und dazu müssen die einzelnen Knoten so lange wie möglich mit ihrem begrenzten Energievorrat auskommen. Um möglichst effizient mit der vorhanden Energie umzugehen, ist auch die Aggregation von Daten als ein Paradigma für Sensornetze erkannt worden. Die Idee dabei ist, die Daten bereits im Netzwerk zu verarbeiten, um die Anzahl der Nachrichten zu minimieren, und somit Energie zu sparen. Dabei kann oft davon profitiert werden, dass verschiedene Sensorknoten das gleiche Phänomen beobachten und ihre Daten dadurch eine hohe Redundanz aufweisen, die dann für die Aggregation ausgenutzt werden kann. Die Aggregationsmöglichkeiten in einem Sensornetz sind allerdings stark von der jeweiligen Anwendung abhängig. Weiter müssen die Sensorknoten „application-aware“ sein, damit ein Sensorknoten, der Daten von verschiedenen Quellen erhält, diese auch richtig aggregieren kann. „Application-aware“ bedeutet dabei, dass ein Sensorknoten wissen muss, welchem Zweck das Sensornetz dient, in dem er sich befindet.

Aufgrund der oben aufgeführten Gründe können die Anforderungen und Ziele abgeleitet werden, die Kommunikationsprotokolle für Sensornetz erfüllen sollten. Als erstes müssen die Protokolle natürlich die Verbreitung von Daten ermöglichen. Zusätzlich sollten sie die Datenaggregation in den Sensorknoten unterstützen. Weiter müssen die Protokolle möglichst energieeffizient sein. Dies impliziert, dass für die Verbreitung der Daten möglichst

wenig Nachrichten benützt werden sollten, da die drahtlose Kommunikation sehr energieintensiv ist. Weil Sensornetze aus mehreren tausend Knoten bestehen können, müssen die Protokolle skalierbar sein. Es sollte also keine zentrale Kontrolle vorhanden sein, und die Protokolle sollten nur lokale Algorithmen verwenden. Und schliesslich sollten sie auch robust gegen Knotenausfälle und das einstreuen neuer Knoten sein.

Im folgenden werden vier speziell für Sensornetze entwickelte Kommunikationsprotokolle vorgestellt, die versuchen die oben genannten Ziele (Energieeffizienz, Skalierbarkeit und Anpassungsfähigkeit) mittels verschiedener Ansätze zu erfüllen.

SPIN

SPIN (Sensor Protocols for Information via Negotiation) [1] ist eine Variante des klassischen Flooding. Wie bei Flooding werden auch bei SPIN die Daten zu allen Knoten im Netz verbreitet. SPIN behebt aber zwei Nachteile von Flooding, nämlich das Implosion- und das Overlap-Problem. Das Implosion-Problem tritt auf, wenn die Knoten die Daten immer an alle Nachbarn weiterleiten, egal ob diese die Daten bereits von einem anderen Nachbarn empfangen haben oder nicht. Das Overlap-Problem entsteht, wenn verschiedene Sensorknoten dasselbe geographische Gebiet abdecken und dadurch redundante Daten sammeln. Anstatt die Daten für das sich überlappende Gebiet nur einmal an einen gemeinsamen Nachbarn zu schicken, senden die Knoten zwei separate Kopien und verschwenden damit wertvolle Energie und Bandbreite.

SPIN bewältigt die beiden oben erwähnten Probleme des klassischen Floodings mit Hilfe eines Handshake-Protokolls. Dabei verhandeln die Knoten vor dem Senden der eigentlichen Daten über deren Inhalt, um sicherzustellen, dass nur neue Daten übermittelt werden. Dazu wird eine Datenbeschreibung, sogenannte Metadaten, verwendet. Das Format dieser Metadaten ist dabei anwendungsspezifisch. Es wird aber angenommen, dass die Metadaten kürzer als die dazugehörigen Daten sind. Weiter soll gelten, dass wenn die Daten unterscheidbar sind, sich auch die dazugehörigen Metadaten unterscheiden, und umgekehrt. Das SPIN Protokoll ist dreistufig, und die Knoten benutzen drei verschiedene Nachrichtentypen um zu kommunizieren:

- ADV - neue Daten anbieten. Wenn ein Knoten neue Daten erhält oder erzeugt, kann er diese Tatsache seinen Nachbarn ankündigen, indem er ihnen eine ADV-Nachricht mit den dazugehörigen Metadaten schickt.
- REQ - Daten anfordern. Ein Knoten sendet eine REQ-Nachricht, wenn er die tatsächlichen Daten erhalten möchte, die mit einer ADV-Nachricht angeboten wurden.
- DATA – eigentliche Daten. Eine DATA-Nachricht enthält die tatsächlichen Daten mit den dazugehörigen Metadaten als Header.

Weil die ADV- und REQ-Nachrichten nur Metadaten enthalten, sind sie um Grössenordnungen kleiner, und verbrauchen beim Senden und Empfangen damit weniger Energie als die eigentliche DATA-Nachricht.

Das Protokoll startet, wenn ein Knoten neue Daten erhält. Um die Daten zu verbreiten sendet er eine ADV-Nachricht, welche die Metadaten enthält, an alle seine Nachbarn. Wenn ein Nachbar an den Daten interessiert ist, sendet er eine REQ-Nachricht und bekommt danach die Daten mit einer DATA-Nachricht zurückgeschickt. Der Knoten wiederholt diese Prozedur mit seinen Nachbarn, und so werden die Daten im ganzen Sensornetz verbreitet (siehe Abb. 1).

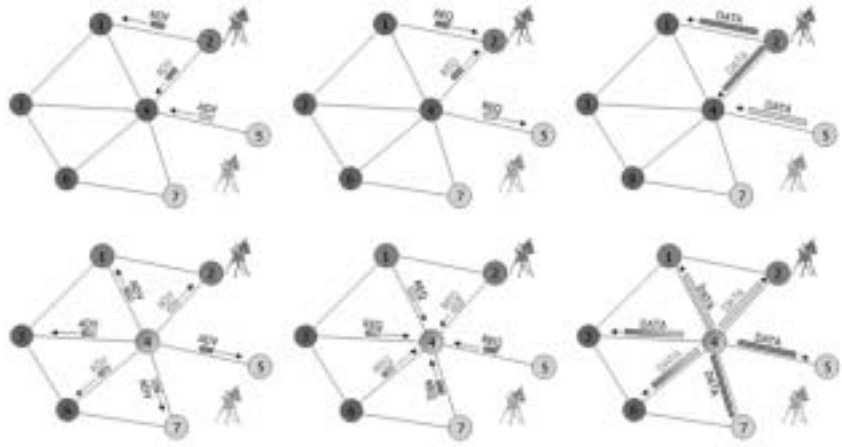

Abbildung 1

Die Vorteile von SPIN liegen in der Einfachheit des Protokolls. Jeder Knoten kommuniziert nur mit seinen Nachbarn, so dass das Protokoll auch bei einer grossen Anzahl von Knoten noch skaliert. Weiter ist SPIN durch das Lösen des Implosion- und Overlap-Problems weitaus energieeffizienter als das klassische Flooding, aber trotzdem immer noch immun gegenüber Veränderungen in der Topologie.
Der Hauptnachteil von SPIN liegt darin, dass das Verfahren nicht geeignet ist, wenn Flooding eigentlich gar nicht benötigt wird. So ist es sehr ineffizient die Daten im ganzen Sensornetz zu verbreiten, obwohl eigentlich nur ein Knoten ganz in der Nähe der Quelle die Informationen benötigt. Der Nachteil gegenüber Flooding ist, dass bei SPIN Metadaten der eigentlichen Daten benötigt werden.

Directed Diffusion

Bei Directed Diffusion [2] handelt es sich um ein gradientenbasiertes Verfahren. Das heisst, es werden Pfade von einer Quelle zur Senke eingerichtet, um die Daten entlang dieser Pfade zu routen. Die Pfadkonstruktion wird dabei von der Senke eingeleitet, so dass die Daten erst bei Bedarf verbreitet werden. Die Daten werden dabei durch eine Menge von Attribut-Wert Paaren repräsentiert.

In Directed Diffusion werden Aufgaben mittels sogenannter Interests im Sensornetz verbreitet. Diese Interest-Verbreitung etabliert die Gradienten im Sensornetz. Wird ein Ereignis, für das ein Knoten Gradienten eingerichtet hat, beobachtet, so beginnen die Daten für dieses Ereignis entlang der Pfade, die von den Gradienten gebildet werden, in Richtung

des Interest-Ursprungs zu fliessen. Das Sensornetz verstärkt einen dieser Pfade, sodass die meisten Daten für ein Ereignis nur auf einem Pfad von der Quelle zur Senke fliessen. Im folgenden werden diese verschiedenen Elemente von Directed Diffusion näher erläutert, und zwar anhand eines Sensornetzes, das die Aufgabe hat, Standorte von Tieren zu bestimmen.

In Directed Diffusion werden Aufgaben mittels einer Liste von Attribut-Wert Paaren beschrieben. Eine Aufgabe in unserem Sensornetz könnte zum Beispiel wie folgt aussehen:

type = bambi
interval = 20 ms
duration = 120 s
rect = [0,200,200,400]

Eine solche Aufgabenbeschreibung spezifiziert eigentlich ein Interesse für Daten, die diesen Attributen entsprechen, deshalb spricht man auch von einem Interest. Im obigen Beispiel will man wissen, ob ein Sensorknoten in einer gewissen geographischen Region ein Bambi beobachtet. Ein Interest hat noch zwei weitere Attribute, das Interval- und das Duration-Attribut. Das Interval-Attribut gibt an, mit welcher Frequenz ein Knoten die Daten zurück zur Senke routen soll, wenn er ein entsprechendes Ereignis beobachtet hat. Das Duration-Attribut gibt an, für wie lange der Knoten die Daten zur Senke senden soll. Benötigt ein Knoten bestimmte Informationen über ein Ereignis, so injiziert er einen Interest ins Sensornetz, der das Ereignis beschreibt. Bei diesem anfänglichen Interest wird das Interval-Attribut sehr gross gewählt. Dies kann als eine Sondierung angesehen werden, in der geprüft wird, ob überhaupt Sensorknoten vorhanden sind, die ein passendes Ereignis registrieren.

Die Verbreitung des Interests kann zum Beispiel mittels Flooding oder Geo-Cast erfolgen. Empfängt ein Knoten einen Interest, so richtet er einen Gradienten dafür ein. Ein Gradient besteht in unserem Beispiel aus zwei Komponenten, nämlich aus einer Datenrate, die durch das Interval-Attribut des Interests festgelegt wird, und aus einer Richtung, hier einfach der Nachbar der den Interest gesendet hat. Anschliessend leitet der Knoten den Interest an eine Teilmenge seiner Nachbarn weiter. Die so im Sensornetz eingerichteten Gradienten bilden Rückwärtspfade von den potentiellen Quellen bis zum Knoten, der den Interest injiziert hat.

Beobachtet ein Knoten ein Ereignis, prüft er nach, ob ein dazu passender Interest gespeichert wurde. Wenn sich der Knoten innerhalb des im Interest spezifizierten Bereichs befindet, beginnt er die Daten, die das Ereignis beschreiben, mit der gewünschten Datenrate zu verbreiten. Die Daten werden dazu an die Nachbarn gesendet, für die der Knoten zuvor Gradienten eingerichtet hat. Zwischenknoten die Daten von einem Nachbarn empfangen, leiten diese entlang der lokal vorhandenen Gradienten weiter. Erhält ein Knoten dabei Daten von mehreren Nachbarn zugesandt, so können diese anwendungsspezifisch aggregiert werden.

Treffen die ersten Daten mit einer kleinen Datenrate bei der Senke ein, verstärkt sie einen Pfad, um qualitativ höherwertige Daten von diesem Ereignis zu erhalten. Bessere Qualität meint hier, dass die Daten mit höherer Frequenz von der Quelle zur Senke geleitet werden. Um einen Pfad zu verstärken, schickt die Senke eine sogenannte Reinforcement-Nachricht an

einen ihrer Nachbarn. Eine Reinforcement-Nachricht ist dabei ein identischer Interest für das Ereignis, mit dem Unterschied, dass nun das Interval-Attribut einen kleineren Wert aufweist. Es gibt dabei verschiedene Regeln, mit denen der Nachbarn ausgewählt wird, der die Reinforcement-Nachricht empfangen soll. Zum Beispiel kann immer der Nachbar „reinforced“ werden, von dem man die neuen Daten zuerst erhält. Wenn ein Knoten eine Reinforcement-Nachricht erhält, so passt er den dazugehörigen Gradienten an und schickt die Nachricht unter Anwendung der gleichen Regel an einen seiner Nachbarn weiter, es sei denn, er erhält von einem seiner Nachbarn die Daten schon mit der geforderten Frequenz. Wird diese Regel rekursiv von allen Zwischenknoten angewandt, die eine Reinforcement-Nachricht erhalten, so wird ein Pfad verstärkt, der eine kleine Latenz aufweist.

Ein Vorteil von Directed Diffusion ist, dass, wenn kontinuierlich Daten von einem Ereignis benötigt werden, durch das Konzept der Reinforcement-Nachrichten, die Daten meistens nur auf einem Pfad von der Quelle zur Senke fliessen. Weiter werden die Nachrichten bei einer Anwendung der oben erwähnten Reinforcement-Strategie sogar auf kürzesten Pfaden geroutet. Ein weiterer Vorteil ist, dass trotzdem mehrere Pfade zwischen Quelle und Senke aufrechterhalten werden, so dass Knotenausfälle bis zu einem gewissen Masse toleriert werden können. Ein Nachteil von Directed Diffusion ist, dass die Gradienten-Setup-Phase energieintensiv ist.

LEACH

LEACH [3] steht für Low-Energy Adaptive Clustering Hierarchy und ist eine Clustering-Variante. Dieses Protokoll geht von einem Sensornetz aus, in dem alle Knoten ihre Daten an eine Basisstation senden wollen. Wie bei konventionellen Clustering-Protokollen wird auch in LEACH das Sensornetz in einzelne Cluster aufgeteilt, und die Knoten in einem Cluster nur mit einem ausgezeichneten Knoten, dem Cluster-Head, kommunizieren. Die Cluster-Heads leiten dann die Daten aller anderen Cluster-Knoten an die Basistation weiter. Weil die Cluster-Heads als Relaisstationen verwendet werden, verbrauchen sie viel Energie. Das LEACH Protokoll versucht nun den Energieverbrauch trotzdem gleichmässig im Sensornetz zu verteilen, indem es die Cluster-Heads nicht fix wählt, sondern sie zufällig rotiert.

LEACH ist dabei in Runden aufgeteilt, wobei sich jede Runde wieder in zwei Phasen aufgliedert. Die Setup-Phase dient dazu, die Cluster zu bilden, und die Cluster-Heads zu bestimmen. In einer Steady-State-Phase werden die Daten anschliessend von den Knoten über die Cluster-Heads zur Basisstation geroutet. Die Steady-State-Phase dauert dabei länger als die Setup-Phase, um den Overhead, der bei der Bildung der Cluster entsteht, möglichst klein zu halten. Im folgenden werden die beiden Phasen von LEACH im Detail besprochen.

Am Anfang jeder neuen Runde werden in der Setup-Phase die Cluster neu gebildet. Jeder Knoten bestimmt dabei für sich, ob er in dieser Runde zu einem Cluster-Head wird. Weil jeder Knoten individuell entscheiden kann, ob er Cluster-Head wird, er also nicht mit seinen Nachbarn kommunizieren muss, benötigt die Wahl der Cluster-Heads wenig Energie. Die Entscheidung basiert dabei auf einem a priori festgelegten Anteil von Knoten, die Cluster-Heads sein sollen, und wie oft ein Knoten bereits Cluster-Head war. Um zu entscheiden ob

ein Knoten *n* Cluster-Head wird, wählt er eine Zufallszahl aus dem Intervall [0,1) und vergleicht diese anschliessend mit einem Schwellenwert T(*n*). Wenn die Zahl kleiner als der Schwellenwert T(*n*) ist, wird der Knoten zu einem Cluster-Head für die aktuelle Runde. T(*n*) wird dabei wie folgt berechnet:

$$T(n) = \begin{cases} \dfrac{P}{1 - P * (r \bmod \frac{1}{P})} & \text{, wenn } n \in G \\ 0 & \text{, sonst} \end{cases}$$

Dabei ist *P* der gewünschte Prozentsatz an Cluster-Heads (z.B. *P* = 0.05), r entspricht der aktuellen Runde und G ist die Menge der Knoten, die in den letzen 1/*P* Runden noch keine Cluster-Heads waren. Wird dieser Schwellenwert benutzt, wird jeder Knoten in 1/*P* Runden genau einmal Cluster-Head.

Sind die Cluster-Heads bestimmt, senden diese eine Advertisement-Nachricht mit einer fest vorgegebenen Sendeleistung aus. Die übrigen Knoten empfangen diese Nachrichten, und entscheiden, welchem Cluster sie in dieser Runde beitreten wollen. Sie antworten dazu dem Cluster-Head, bei dessen Advertisement-Nachricht die höchste Signalstärke gemessen wurde. Geht man nämlich von symmetrischen Kanälen aus, erreicht ein Knoten diesen Cluster-Head mit der geringsten Sendeleistung. Der Cluster-Head erstellt nun einen TDMA-Schedule, in dem jedem Knoten im Cluster ein festes Zeitfenster zur Kommunikation mit dem Cluster-Head zugeteilt wird. Weiter wählt er einen CDMA-Code zur Codierung der Bits. Da jedes Cluster einen anderen CDMA-Code wählt, kann innerhalb der Cluster gleichzeitig kommuniziert werden, ohne dass sich zwei Cluster gegenseitig stören.

Sind die Cluster gebildet und die TDMA-Schedules an die Knoten verteilt, beginnt die Steady-State-Phase. Die Knoten senden jetzt ihre Daten in dem für sie reservierten Zeitfenster an den Cluster-Head. Dabei nutzen sie das Wissen über die Signalstärke der Advertisement-Nachricht, um mit der minimalen Energie senden zu können. Nachdem der Cluster-Head von jedem Knoten im Cluster Daten erhalten hat, aggregiert er diese und sendet das Resultat anschliessend an die Basisstation. Weil diese weit entfernt ist, muss diese Übertragung mit hoher Energie erfolgen.

Die Vorteile von LEACH sind, dass durch die Rotation der Cluster-Heads der Energieverbrauch gleichmässig über die Sensorknoten verteilt wird, und dass die Cluster-Knoten in der Steady-State-Phase auf Grund des TDMA-Modus ihr Kommunikationsmodul über weite Strecken abschalten können. Ein weiterer Vorteil ist auch, dass diese Clustering-Idee mehrstufig angewandt werden kann, was zu einem hierarchischen Clustering führt. Ein Nachteil von LEACH liegt darin, dass implizit angenommen wird, dass alle Knoten die Basisstation erreichen. Für die meisten Sensornetze gilt diese Annahme aber nicht. Ein weiterer Nachteil dieses Protokolls ist auch, dass bei einem Ausfall eines Cluster-Heads, die Basisstation für eine Runde von allen Knoten im Cluster keine Daten mehr erhält.

TTDD

TTDD [4] steht für Two-Tier Data Dissemination. Dieses Protokoll wurde speziell dazu entwickelt, um mobile Senken in einem Sensornetz zu unterstützen. Eine Senke ist dabei ein Gerät, von dem aus Anfragen an das Sensornetz gestellt werden, um Daten aus diesem zu extrahieren. Die Knoten des Sensornetzes selbst sollen dabei stationär sein und über Standortkenntnisse verfügen.

Bei TTDD baut ein Knoten, der ein Ereignis beobachtet hat, eine Infrastruktur für die Verbreitung der Daten auf. Dies wird gemacht, um zu verhindern, dass eine potenzielle Senke für eine Anfrage das ganze Sensornetz fluten muss. Dazu wird von der Quelle ein Gitternetz über das ganze Sensornetz aufgebaut (siehe Abb. 2).

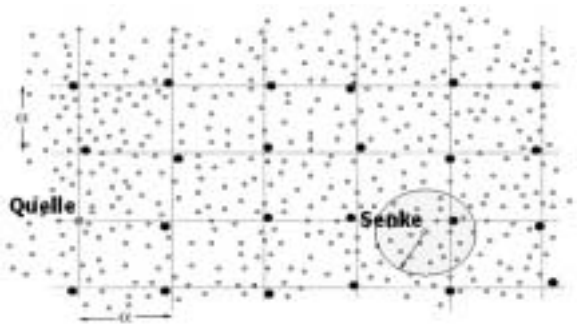

Abbildung 2

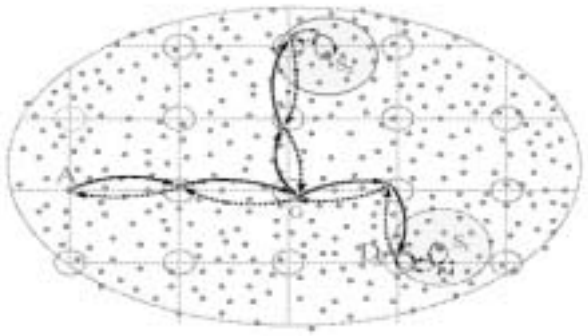

Abbildung 3

Der Parameter α, der die Grösse der einzelnen Zellen des Gitters bestimmt, ist dabei fest vorgegeben. Um das Gitter zu konstruieren, berechnet die Quelle aus ihrer Position und α seine vier benachbarten Gitterpunkte. Anschliessend sendet sie vier Nachrichten mittels Greedy-Geographical-Forwarding an diese geographischen Punkte. Greedy-Geographical-Forwarding funktioniert dabei so, dass ein Knoten, der eine solche Nachricht empfängt, diese einfach an den Nachbarn weiterleitet, dessen Position am nächsten beim berechneten Punkt liegt. Gibt es keinen solchen Nachbarn, und ist der Knoten näher am berechneten Punkt als α/2 so wird er zu einem Gitterknoten und propagiert die Nachricht an seine drei benachbarten Gitterpunkte weiter. Ist er aber weiter entfernt als α/2 so macht er gar nichts. Dieses Verhalten garantiert, dass die Gitterkonstruktion bei den Rändern des Sensornetzes korrekt terminiert.

Bei der Datenverbreitung gibt es, wie der Name des Protokolls schon sagt, zwei Schichten. Die erste Schicht beschreibt die Datenverbreitung innerhalb einer Zelle des Gitters. Die zweite Schicht umfasst die Datenverbreitung entlang des Gitters. Benötigt eine Senke bestimmte Daten, so führt sie nur ein lokales Flooding aus. Wird eine solche Anfrage von einem Gitterknoten empfangen (dem unmittelbaren Gitterknoten für diese Anfrage), wird diese wieder mittels Greedy-Geographical-Forwarding entlang des Gitters zur Quelle geleitet. Dabei merkt sich jeder Gitterknoten auf dem Weg zur Quelle seinen Vorgänger, damit die Daten von der Quelle wieder den gleichen Weg durch das Gitter bis zum unmittelbaren Gitterknoten nehmen. Erhält ein Gitterknoten eine Anfrage durch den bereits

Daten fliessen, so sendet er die Anfrage nicht weiter, sondern leitet direkt eine Kopie der Daten an den Gitterknoten weiter, von dem er die Anfrage erhalten hat.

Sind die Daten beim unmittelbaren Gitterknoten angelangt, dann werden sie mit dem sogenannten Trajectory-Forwarding zur Senke weitergeleitet. Trajectory-Forwarding wird dabei angewandt um explizit die Mobilität der Senke zu berücksichtigen. Weil angenommen wird, dass sich die Senke kontinuierlich bewegt, und ihren Standort nicht kennt, wird sie bei diesem Verfahren von zwei Sensorknoten, den sogenannten Agenten, vertreten. Der primary-Agent repräsentiert die Senke für den unmittelbaren Gitterknoten. Das heisst, der unmittelbare Gitterknoten kennt den Standort der Senke gar nicht, sondern er leitet die empfangenen Daten einfach an den primary-Agenten weiter, und dieser routet die Daten dann zum Immediate-Agenten. Der immediate-Agent ist dabei immer ein Knoten, der nur einen Hop von der Senke entfernt ist. Er wird benötigt, weil eine Senke ihren genauen Standort nicht kennt, die Daten aber aufgrund von Ortinformationen geroutet werden. Abbildung 3 zeigt noch einmal wie die Daten von einer Quelle (A) zu zwei Senken (S_1 und S_2) verbreitet werden.

Die Vorteilte von Two-Tier Data Dissemination sind, dass mehrere Anfragen in einem Gitterknoten aggregiert werden können, und dass das Flooding der Anfragen nur lokal erfolgen muss. Ein weiterer Vorteil von TTDD ist, dass zudem explizit mobile Senken unterstützt werden. Ein Nachteil dieses Protokolls ist der hohe Energieverbrauch bei der Konstruktion des Gitters. Weiter ist zu sagen, dass TTDD bei vielen und oft wechselnden Quellen schlecht skaliert, weil die Gitterkonstruktion viel Energie benötigt.

Fazit

Der Bericht stellt vier verschiedene Protokolle zur Datenverbreitung in Sensornetzen vor. Eine abschliessende Bewertung der vier Protokolle ist nicht möglich, denn die Protokolle beruhen einerseits auf unterschiedlichen Annahmen, was die Ausprägungen des Sensornetzes betreffen, und andererseits gibt es noch keine vergleichenden Messungen. Da die Energie in Sensornetzen ein kritischer Faktor ist, müssen die Kommunikationsprotokolle sehr genau auf die jeweilige Anwendung und das zugrundeliegende Sensornetz angepasst werden. Es ist also zu erwarten, dass sich nicht ein Protokoll durchsetzen wird, sondern dass für verschiedene Anwendungen verschiedene Protokolle für die Datenverbreitung eingesetzt werden.
Die vier vorgestellten Ansätze sollten die Breite der Ansätze dokumentieren, die beim Entwurf von Protokollen zur Datenverbreitung in Sensornetzen vorliegt. So haben wir mit SPIN und LEACH zwei Verfahren kennen gelernt, bei denen die Datenverbreitung von der Quelle initiiert wird, während in Directed Diffusion und TTDD erst eine Interessensbekundung der Senke zur Verbreitung der Daten führte. Weiter gibt es auch Unterschiede was die Unterstützung der Knotenmobilität angeht. So werden bei TTDD explizit mobile Senken unterstützt, wohingegen das LEACH-Protokoll von einem stationären Sensornetz ausgeht.

Quellen

[1] Kulik, Heinzelman, Balakrishnan, „Negotiation-based Protocols for Disseminating Information in Wireless Sensor Networks“, Wireless Networks, Vol. 8, 2002, pp. 169-185

[2] Intanagonwiwat, Govindan, Estrin, „Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks“, Proc. ACM MobiCom ´00, Boston, August 2000

[3] Heinzelman, Chandrakasan, Balakrishnan, „Energy-Efficient Communication Protocol for Wireless Microsensor Networks“, Proc. HICSS ´00, January 2000

[4] Ye, Luo, Cheng, Lu, Zhang, „A Two-Tier Data Dissemination Model for Large-scale Wireless Sensor Networks“, Proc ACM/IEEE MobiCom ´02, Atlanta, September 2002