Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Marc Cortesi

Energy Efficient Deployment of Wireless Sensor Networks

Student Thesis SA-2005-12
Winter Term 2004/2005

Tutor: Pascal von Rickenbach

Supervisor:
Prof. Roger Wattenhofer

28.2.2005

Institut für Technische Informatik und Kommunikationsnetze
Computer Engineering and Networks Laboratory

1 Zusammenfassung

In dieser Arbeit werden zwei Probleme von statischen drahtlosen Sensornetzwerken behandelt. Einerseits betrachteten wir Methoden um Energie zu sparen während einer langen Deployment-Phase, andererseits untersuchten wir effizientes Aufwecken aller Knoten zu Beginn der Discovery-Phase. Zu diesem Zweck werden zwei randomisierte Algorithmen genauer betrachtet. Beide nutzen zufällige und unabhängige Kommunikation um innert kürzester Zeit möglichst alle Knoten im Netzwerk zu erreichen und aufzuwecken, auch bekannt als das Wake-Up Problem.
Für den Birthday Algorithmus [1] analysierten wir die Dauer bis alle Knoten informiert sind im average-case. Im weiteren wurden die beiden Algorithmen gegeneinander getestet und es konnte empirisch gezeigt werden, dass der Probability Increase Algorithmus [2] im Mittel schneller alle Knoten im Netzwerk erreicht, dies bei gleicher Energieersparnis in der Deployment-Phase.

2 Einleitung

Entscheidende Fortschritte auf den Gebieten der kabellosen Kommunikation und der Mikroelektronik haben zur Vision der Sensornetzwerke geführt. Bestehend aus einer sehr grossen Zahl von kleinen, leichten und billigen Knoten, jeder davon ausgestattet mit Prozessor, Batterie, Speicher und einer Funkschnittstelle, erhofft man sich viele neue Einsatzgebiete zu erschliessen. Solche Netzwerke könnten in einer realen Umgebung autonom Information sammeln und teilweise bereits verarbeiten. Speziell in für den Menschen schwer zugänglichen oder gefährlichen Gebieten wäre diese neue Möglichkeit der Überwachung physikalischer Phänomene äusserst interessant. Dadurch dass die Knoten komplett autark sein müssen und nur mit einer kleinen Batterie oder einer schwachen Solarzelle ausgestattet sind, rückt der effiziente Energieverbrauch ins Zentrum des Forschungsinteresses.
Oft ist die Struktur des Netzes unvorhersehbar, d.h. die Knoten haben keinerlei Information über ihre topologische Beziehung zu anderen Knoten. Bei einem möglichen Abwurf aus einem Flugzeug wäre das der Fall. Diese erste Phase der Installation des Sensornetzwerkes nennt man Deployment. Energieersparnis ist in solchen Netzwerken von entscheidender Wichtigkeit. Einerseits möchte man natürlich erreichen, dass das Sensornetzwerk möglichst lange seine Aufgabe erfüllen kann. Andererseits hätte ein Ausfall eines Knotens unter Umständen das Verschwinden des von ihm unterstützten Datenpfades zur Folge. Deshalb versucht man zu erreichen, dass die Knoten möglichst schlafen solange nichts Entscheidendes passiert. Schlafen in diesem Zusammenhang bedeutet, dass die drahtlose Kommunikationsmöglichkeit nicht genutzt wird. Trotzdem muss das Netzwerk unter Umständen sehr

rasch reagieren können, sollte solch ein Ereignis schlussendlich eintreffen. Das kann beispielsweise die Messung eines Sensorwertes sein oder die Initialisierung der Discovery bei einer etappenweisen Ausstreuung der Knoten. Nach diesem Zustandsübergang möchte man erreichen, dass die Sensorknoten möglichst alle ihre Nachbarn entdecken, deshalb der Name Discovery.
Um während der Deployment-Phase Energie zu sparen, wird das Funkmodul der Sensorknoten nur sporadisch eingeschaltet damit man hört ob jemand sendet. Dabei wird der Zustand für jeden Zeitslot unabhängig und mittels einer Wahrscheinlichkeitsverteilung bestimmt. Ein Knoten sendet nur dann, wenn er bereits die Discovery-Phase erreicht hat. Geht einer oder mehrere Knoten in diese Phase über, so ist es das Ziel in kürzester Zeit möglichst alle Knoten im Netzwerk davon zu unterrichten. Die informierten Knoten beginnen nun ebenfalls zu senden und versuchen die Nachricht an ihre Nachbarn weiterzupropagieren. Nachdem hoffentlich alle verfügbaren Knoten aufgeweckt worden sind, könnte damit begonnen werden eine intelligente Routing-Struktur aufzubauen (Clustering, Source Routing etc.). Damit würde der Datenaustausch während der operativen Phase des Sensornetzwerkes wesentlich erleichtert. Damit habe ich mich aber im Rahmen meiner Semesterarbeit nicht beschäftigt.

3 Motivation

Heutige kommerzielle drahtlose Systeme haben keine Möglichkeit eines speziellen Aufwachsignales. Dies bedeutet, dass Knoten von Zeit zu Zeit ins Netz hineinhören müssen um mit den Nachbarn in Kontakt zu bleiben. Das Zuhören kostet Energie und zwar von der Grössenordnung die auch das Senden benötigen würde. Dies trifft insbesondere für drahtlose Systeme zu, die mit einer geringen Leistung senden (vgl. [5]) und genau diese sind für unsere Zwecke interessant. Ohne die Fähigkeit aufgeweckt zu werden, gäbe es nur noch zwei Möglichkeiten Übertragungsenergie zu sparen: mit niedrigerer Leistung zu senden oder ganz aufs Senden zu verzichten. Was das erste Gebiet betrifft, gibt es schon sehr viele Resultate, die unter dem Begriff Topologiekontrolle zusammengefasst werden (vgl. z.B. [3]). Wir beschäftigen uns hier mit der zweiten Variante indem ein Sensorknoten jeweils nur mit einer bestimmten Wahrscheinlichkeit sendet oder horcht und andernfalls schläft.

4 Modell

In diesem Kapitel definieren wir das Modell welches Analyse und Simulation zugrundeliegt. Dabei wurde darauf geachtet sowohl mathematisch präzise Resultate zu erreichen als auch die Realität möglichst genau abzubilden.
Wir betrachten für die Analyse des Birthday Protokolls die Idealisierung eines single-hop Netzwerkes, d.h. potentiell kann jeder Knoten jeden anderen

erreichen. Für die Simulation haben wir den allgemeineren multi-hop Fall vorausgesetzt, in dem die Restriktion auf einen kompletten Graphen fallengelassen wurde.
Zum Modellieren des Netzwerkes wird der bekannten Unit Disk Graph (UDG) herangezogen. In einem UDG $G = (V, E)$ gibt es genau dann eine Kante zwischen zwei Knoten u und v wenn die euklidische Distanz der beiden höchstens 1 beträgt.
Knoten haben kein a priori Wissen über die topologische Anordnung des gesamten Netzwerkes, d.h. auch nicht über sich selbst relativ zu ihrer unmittelbaren Nachbarschaft. Als Anhaltspunkt gilt einzig eine obere Schranke $\hat{n}$ für die totale Anzahl Knoten $n = |V|$. Da Knoten ausfallen können, variiert n. Der Wert $\hat{n}$ ist aber über alle Knoten konsistent.
Die Zeit wird in konstant lange Zeitslots aufgeteilt. Um die Analyse zu vereinfachen, nehmen wir an, dass die Knoten synchronisiert sind. In [4] wurde gezeigt, dass sich diese Idealisierung nur um Faktor 2 vom realistischen asynchronen Fall unterscheidet wo die Knoten zwar unabhängig voneinander die Zeit in slots aufteilen aber mit der selben Frequenz. Intuitiv lässt sich dies folgendermassen erklären: Ein einzelnes in einem slot gesendetes Paket kann nicht mit mehr als zwei aufeinanderfolgenden slots eines anderen Knoten interferieren.
Den Sensorknoten steht keine Kollisionserkennung zur Verfügung. Speziell können sie nicht unterscheiden ob mehrere benachbarte Knoten senden oder ob keiner sendet. Diese Annahme ist realistisch da Sensorknoten mit minimaler Hardware ausgestattet sind, weil sie energieeffizient, leicht und billig sein müssen. Das Netzwerk ist statisch. Nachdem sie in der Deployment-Phase ihren Bestimmungsort erreicht haben, bewegen sich die einzelnen Knoten nicht mehr von der Stelle.

5 Algorithmen

Die hier vorgestellten Algorithmen stammen aus [1] und [2] wobei letzterer für das hier betrachtete Problem adaptiert wurde. Jeder wird auf den Sensorknoten gestartet bevor diese ihren Einsatzort erreicht haben. In einer ersten Phase warten die Knoten auf Nachrichten ohne selbst zu senden. Denkbar wäre nun, dass die Kommunikation im Netz entweder von ausserhalb des Sensornetzwerkes oder aber auch von einem Knoten innerhalb initialisert wird.
In jedem Zeitslot kann ein Knoten entweder senden, zuhören oder schlafen je nachdem in welcher Phase er sich befindet. Sollte er zuhören, so erhält er eine Nachricht genau dann wenn in seiner Nachbarschaft genau ein anderer Knoten im selben slot sendet. In der Deployment-Phase hört ein Knoten entweder oder er schläft. Geht er in die Discovery-Phase über (d.h. er registriert einen entscheidenden Event oder erhält eine entsprechende Nach-

richt), so wird er entweder senden oder zuhören. Dies führt unweigerlich zu einem hohen Energieverbrauch, weswegen man daran interessiert ist diese Phase kurz zu halten.
In beiden Phasen wird der Zustand für jeden einzelnen slot mittels einer zugrundeliegende Wahrscheinlichkeitsverteilung ermittelt. Während des bei beiden Algorithmen identischen Deployments hört ein Knoten mit Wahrscheinlichkeit p_l und dementsprechend schläft er mit Wahrscheinlichkeit $p_s = 1 - p_l$. Analog sendet ein Knoten während der Discovery mit Wahrscheinlichkeit p_t und hört mit der Wahrscheinlichkeit $p_l = 1 - p_t$ zu. Demnach spart ein Knoten den Anteil p_s der Energie während des Deployments.
Die Festsetzung der Wahrscheinlichkeit p_t macht nun den Unterschied zwischen den beiden Algorithmen aus. Während sie beim Birthday Algorithmus über die gesamte Discovery den konstanten Wert $\frac{1}{\hat{n}}$ annimmt, wächst sie beim Probability Increase Algorithmus mit fortschreitender Dauer exponentiell an. Diese Tatsache erklärt auch die Namensgebung. Zu Beginn der Discovery wird die Sendewahrscheinlichkeit mit $\frac{1}{2\hat{n}}$ halb so gross wie beim Birthday Algorithmus gesetzt. Nachdem über eine in n logarithmische Anzahl von inneren Schleifendurchläufen mit dieser Wahrscheinlichkeit gesendet wurde, verdoppelt man sie. Dies wird wiederum logarithmisch in n oft wiederholt. Die Idee dieses Vorgehens ist das Herantasten an die Dichte in der Nachbarschaft eines Knotens. Wenn die angebrachte Sendewahrscheinlichkeit erreicht ist, möchte man innerhalb einer inneren Schlaufe alle Nachbarn erreichen.
Die Anzahl der Knoten, die ausgestreut werden, beträgt $\hat{n}$. Dies ist eine obere Schranke für die tatsächliche Grösse n des Netzwerkes, denn es kann immer wieder passieren dass Knoten beschädigt oder ausserhalb des Empfangsbereiches der anderen zu liegen kommen. Arbeiten zu Slotted Aloha (vgl. [4]) haben gezeigt, dass es besser ist die Anzahl der Nachbarn zu überschätzen als sie zu unterschätzen, da Kollisionen die Kommunikation stark beeinträchtigen können. In [1] wird für die Konstante M im Birthday Algorithmus der Wert 3000 gewählt. Die Länge eines while-Schleifendurchlaufs entspricht exakt der eines slots.

Algorithm 1 Birthday

```
1: informed = false;
2: while not informed and no event do
3:    listen with probability p_l else sleep;
4:    if message received then
5:       informed = true;
6:    end if
7: end while
8: for M slots do
9:    send notification message with probability 1/n̂;
10: end for
```

Algorithm 2 Probability Increase

```
1: informed = false;
2: while not informed and no event do
3:    listen with probability p_l else sleep;
4:    if message received then
5:       informed = true;
6:    end if
7: end while
8: p_t = 1/(2n̂);
9: for 1 to ⌈log(n̂)⌉ + 1 do
10:   for ⌈(c/p_l) · (⌈log(n̂)⌉ + 1)⌉ slots do
11:      send notification message with probability p_t;
12:   end for
13:   p_t = 2p_t;
14: end for
```

6 Analyse

Dieser Abschnitt enthält den theoretischen Teil meiner Arbeit. Aufgrund der Komplexität der Probleme ist er leider nicht allzu umfangreich geworden. Als die Arbeit ins Stocken geriet, haben wir uns zur Simulation der Algorithmen entschlossen.
Da wir sehr stark daran interessiert waren wie sich der Birthday Algorithmus in der Praxis verhalten würde, haben wir eine average-case Analyse der Anzahl slots bis alle n Knoten informiert sind durchgeführt, d.h. wir ermittelten den Erwartungswert der Anzahl slots von der Initialisierung der Discovery-Phase bis zum Zeitpunkt wo der letzte Knoten aufgeweckt wird.
Der Einfachheit halber nehmen wir an, dass $n = \hat{n}$ ist und der zugrundeliegende Graph komplett sei, d.h. jeder Knoten kann jeden anderen hören. T

sei die Menge aller Knoten, die bereits in der Discovery-Phase sind. Dann gilt dass

$$E[\sharp \text{ horchende Knoten pro slot}] = (n - |T|)p_l$$

Die Anzahl der Knoten im Deployment, die pro slot zuhören, ist binomialverteilt $\mathcal{B}(n - |T|; p_l)$. Ein Knoten wird nun informiert genau dann wenn genau ein Knoten sendet.

$$\begin{aligned} E[\sharp \text{ informierter Knoten pro slot}] &= \underbrace{|T|\frac{1}{n}\Big(1 - \frac{1}{n}\Big)^{|T|-1}}_{\Pr[\textit{genau einer sendet}]}(n - |T|)p_l \\ \\ E[\sharp \text{ slots bis ein Knoten informiert}] &= \frac{1}{E[\sharp \text{ informierter Knoten pro slot}]} \\ \\ &= \frac{1}{|T|\frac{1}{n}(1 - \frac{1}{n})^{|T|-1}(n - |T|)p_l} \\ \\ &=: E[R_{|T|}] \end{aligned}$$

$R_{|T|}$ ist nun die Anzahl der slots bis ein Knoten in Discovery übergeht wenn $|T|$ Knoten diese bereits erreicht haben. Analog zum Coupon Collector Problem summieren wir nun alle $R_{|T|}$ für $|T| = 1 \dots n-1$ auf und bestimmen den Erwartungswert dieser Summe.

$$\begin{aligned} E[\sum_i R_i] &= \sum_i E[R_i] = \sum_{i=1}^{n-1} \frac{1}{i\frac{1}{n}(1 - \frac{1}{n})^{i-1}(n-i)p_l} \\ \\ &= n\,\frac{1}{p_l}\sum_{i=1}^{n-1} \frac{1}{\underbrace{(1 - \frac{1}{n})^{i-1}}_{>\frac{1}{e}\ \forall 1 \le i \le n-1}\, i(n-i)} < e\,n\,\frac{1}{p_l}\sum_{i=1}^{n-1}\frac{1}{i(n-i)} \\ \\ &< e\,n\,\frac{1}{p_l}\sum_{i=1}^{n-1}\frac{1}{n-i} = e\,n\,\frac{1}{p_l}\underbrace{\sum_{i=1}^{n-1}\frac{1}{i}}_{O(\log(n))} = O(n\log(n)) \end{aligned}$$

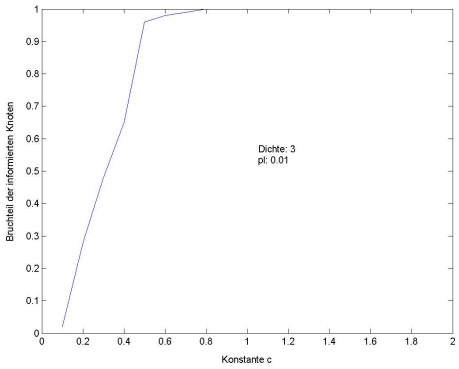
$$\Rightarrow \quad E[\sharp \text{ slots bis alle Knoten informiert}] \;=\; O(n\log(n))$$

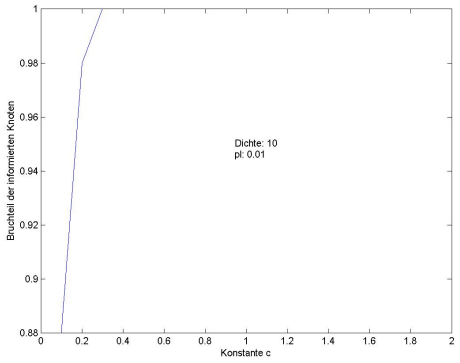
Damit haben wir gezeigt, dass der Birthday Algorithmus in einem kompletten Graphen im Erwartungswert O($n\log(n)$) Zeitslots benötigt um alle n Knoten zu informieren.

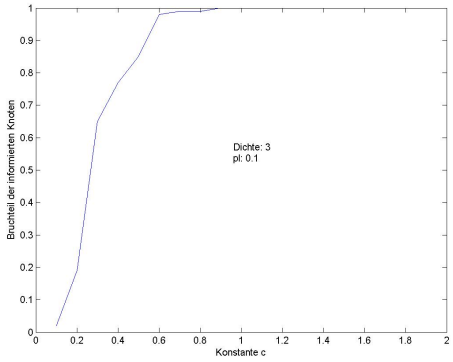
7 Simulation

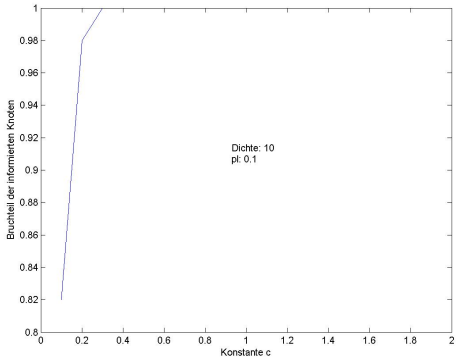
Wiederum waren wir daran interessiert wie lange es dauert bis die Information, dass die Discovery-Phase beginnt, alle Knoten im Netzwerk erreicht hat. Andere interessante Fragen wie ob ein Knoten alle seine Nachbarn kennengelernt hat, haben wir im Rahmen dieser Arbeit nicht betrachtet. Ziel unserer umfangreichen Simulation war es zu zeigen, ob der Probability Increase Algortihmus eine wirkliche Verbesserung gegenüber dem Birthday Algorithmus bringt. Als zweidimensionale Testumgebung diente ein Quadrat mit Seitenlänge 10. Innerhalb dieses Quadrats wurden, je nach Dichte des Graphen, unabhängig gleichverteilt die Knoten angeordnet. Wir haben für die Simulation nur zusammenhängende Graphen verwendet. Dabei variierten wir die Dichte und die Wahrscheinlichkeit p_l dass ein Knoten zuhört.
Als erstes galt es die Grösse der Konstante c im Probability Increase Algorithmus zu ermitteln. Je grösser man c wählt, desto grösser wird die Wahrscheinlichkeit dass alle Knoten informiert werden. Dies sieht man recht leicht, da c direkt proportional zur Anzahl innerer Schleifendurchläufe im Probability Increase Algorithmus ist.
Die folgenden Bilder zeigen den durchschnittlichen Bruchteil aller Knoten die informiert werden konnten, und damit in die Discovery-Phase übergingen, für verschiedene Werte der Konstante c und Netzwerkdichten von 3,10 und 30. Man sieht, dass für geringe Werte von p_l bereits ein kleines c ausreichen würde, so z.B. 1. Da aber verschiedenste Einsatzmöglichkeiten mit unterschiedlichen p_l abgedeckt werden müssen, haben wir uns dazu entschlossen einen höheren Wert für die weiteren Betrachtungen zu verwenden. Grosse Werte für c führen dazu, dass ein Knoten länger sendet und deshalb die Wahrscheinlichkeit von Kollisionen wächst. Dies hat unmittelbare Auswirkungen auf die Laufzeit des Algorithmus. Zudem verbraucht ein Knoten unter Umständen unnötigerweise Sendeenergie.
Bei der Wahl von c kommt es zu einem trade-off zwischen der Wahrscheinlichkeit alle informieren zu können und der Laufzeit bis dies erreicht ist. Da wir für alle Dichten und Wahrscheinlichkeiten p_l ein konsistentes c haben wollten und p_l Priorität beimassen, haben wir uns dazu entschieden c grosszügig zu wählen.
Aus den nun folgenden Bildern wird klar dass, wenn wir $c = 4$ setzen, wir für alle Dichten und p_l ein high-probability Resultat erhalten. Zudem zeigt

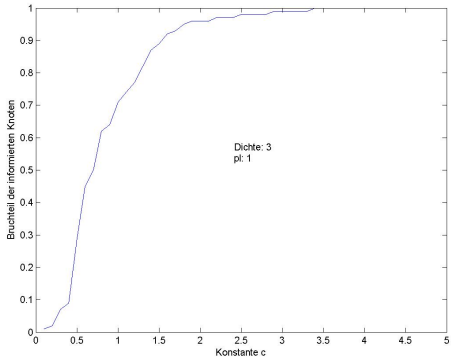
die darauffolgende Tabelle dass die Laufzeitverluste durch die grosszügige Wahl von c nicht frappant sind.

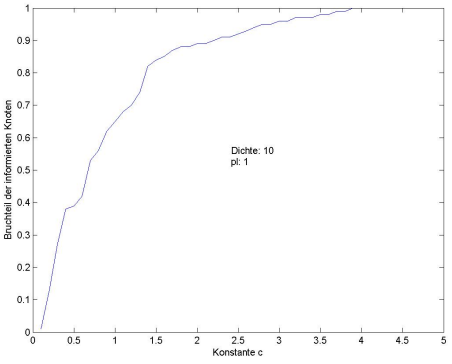


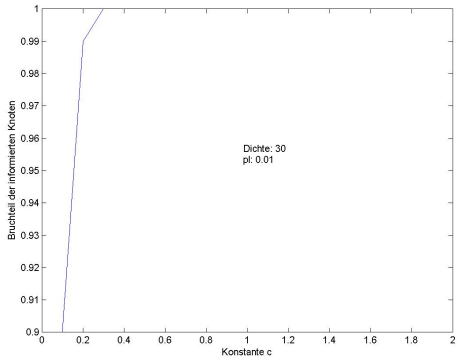


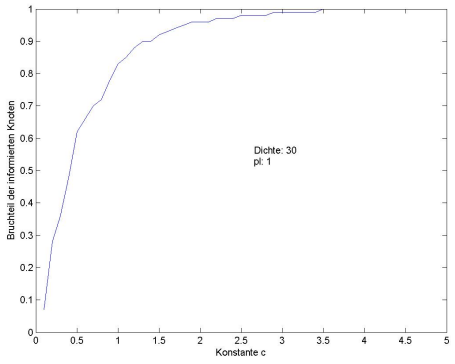


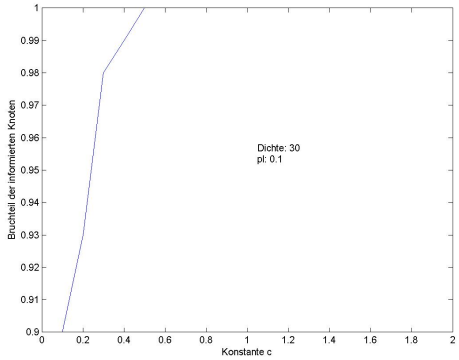












Dadurch dass wir ein c gewählt haben, mit dem wir für alle p_l mit hoher Wahrscheinlichkeit alle Knoten erreicht werden, gehen wir für $p_l = 0.01$ und $p_l = 0.1$ folgende Laufzeitverluste ein:

	3	10	30
0.01	49912 113360 56%	39352 100133 61%	46521 108242 57%
0.1	5403 11318 52%	4660 10174 54%	5145 11255 54%

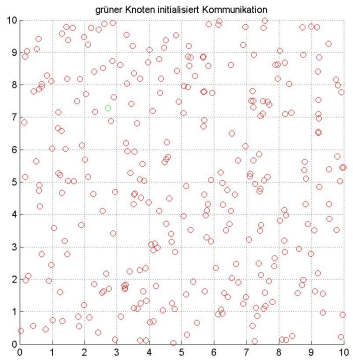
In der obigen Tabelle sind die Simulationsresultate der beiden Algorithmen für die Dichten 3, 10, 30 und $p_l = 0.01$ rsp. 0.1 aufgelistet. In den Feldern mit drei Werten, bedeutet der obere die durchschnittliche Anzahl slots mit $c = 1$, der mittlere dieselbe mit $c = 4$ und der unterste die relative Differenz der beiden. Da wir mit $p_l = 1$ und $c = 1$ mit grosser Wahrscheinlichkeit

nicht alle Knoten hätten benachrichtigen können, erfahren wir hier keinen Laufzeitverlust.
Als nächstes gingen wir dazu über die beiden Algorithmen gegeneinander zu testen. Wir setzten dabei die Konstante $M = \infty$, d.h. im Birthday Algorithmus verblieb ein Knoten im Discovery-Modus bis zum Ende der Simulation. Durch diese Vereinfachung hat jeder Knoten auch noch nach Ablauf der eigentlichen Frist die Möglichkeit Nachbarn zu informieren. Daraus ergibt sich ein Vorteil für den Brithday Algorithmus.
Aus der untenstehenden Tabelle wird ersichtlich, dass der Birthday Algorithmus trotzdem schlechter abschneidet und es deshalb auch mit jedem beliebigen M tun würde. Für jede Simulation instantiierten wir 1000 zufällige Graphen und liessen beide Algorithmen darauf laufen.

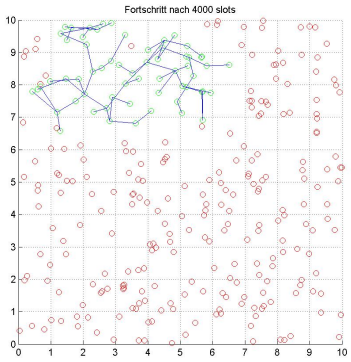
	3	10	30
0.01	164946 112994 31%	109961 103973 5%	110519 106253 4%
0.1	16768 11022 34%	12735 10578 17%	13158 10979 17%
1	2398 1252 48%	2848 1243 56%	4019 1432 64%

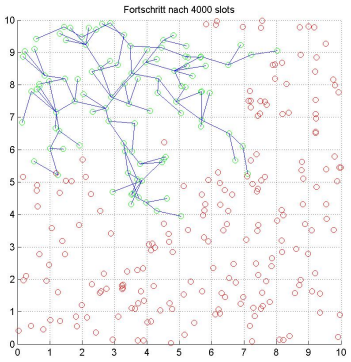
Die erste Zahl steht für die durchschnittliche Anzahl slots des Birthday Algorithmus, die zweite für die des Probability Increase Algorithmus. Die dritte Zahl wiederum ist die relative Differenz der beiden.
Der grosse Unterschied bei $p_l = 1$ ergibt sich durch die Wahl von c. Durch die spezifische Festsetzung von c ist der Probability Increase Algorithmus entscheidend besser. Ebenfalls sticht die um einen Drittel bessere Laufzeit bei der geringen Dichte von 3 hervor.
Zusammenfassend lässt sich sagen, dass der Probability Increase Algorithmus für alle getesteten Dichten und Wahrscheinlichkeiten p_l schneller alle Knoten informierte als der Probability Increase Algorithmus, teils sogar deutlich.

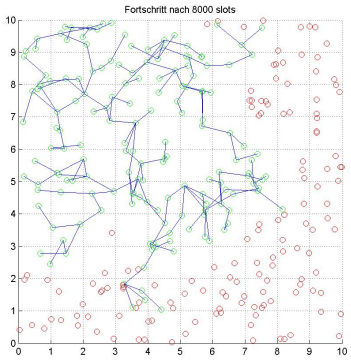
Zum Schluss wird noch ein Beispiel einer Propagierung der Aufwecknachricht der beiden Algorithmen gezeigt. In den folgenden Bildern wird jeder Knoten als Kreis dargestellt. Ein grüner Kreis bedeutet, dass dieser Knoten bereits informiert worden ist und er darauf in die Discovery gewechselt hat. Die roten haben noch nichts gehört und sind demzufolge noch in der Deployment-Phase. Für jeden Knoten wird eine Kante zu jenem Knoten gezeichnet, von dem er die Nachricht zuerst erhalten hat.
Bei beiden Algorithmen fängt der selbe Knoten zu senden an:

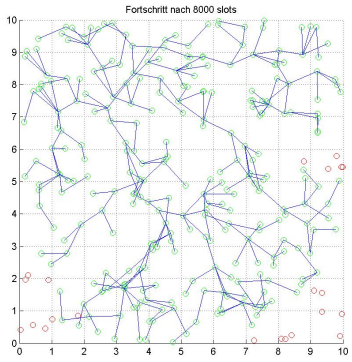


In der Folge sieht man links die Propagierung des Birthday Algorithmus, rechts die des Probability Increase Algorithmus:

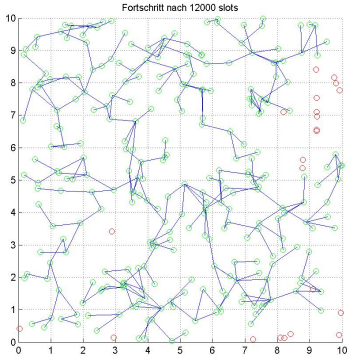


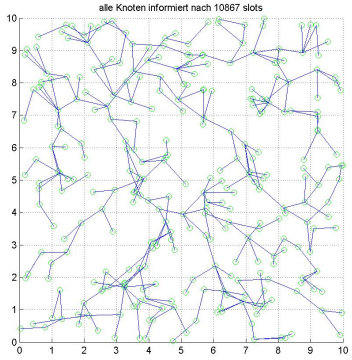




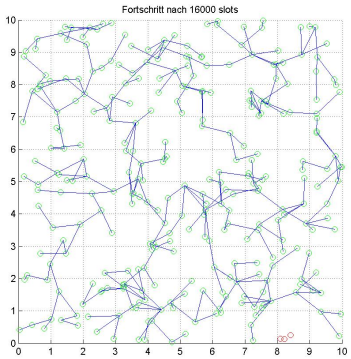


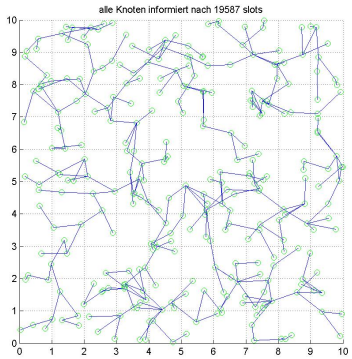
Während der Probability Increase Algorithmus bereits nach 10867 slots alle Knoten erreichen konnte, sieht man beim Birthday Algorithmus noch ein paar Knoten, die noch nicht informiert worden sind.





Die letzten beiden Bilder sind beides Konstellationen des Birthday Algorithmus. Um die wenigen übriggebliebenen Knoten noch erreichen zu können, benötigt er zusätzliche 7587 slots, so dass nach schlussendlich 19587 slots alle Knoten aufgeweckt worden sind. Folglich war der Probability Increase Algorithmus fast doppelt so schnell wie der Birthday Algorithmus.





8 Fazit

Sensornetzwerke sind ein spannendes aktuelles Forschungsgebiet. Es tauchen viele neue Fragestellungen auf oder bekannte werden unter neuen Gesichtspunkten betrachtet. Energieeffizienz wird meiner Meinung nach über längere Zeit ein entscheidender Faktor sein, da die Batteriekapazitäten von allen Ressourcen am wenigsten Moore's Law gehorchen.
Wir haben gesehen, dass randomisierte Kommunikationsprotokolle gegenüber deterministischen Ansätzen mit ihrer Einfachheit und Robustheit in unvohergesehenen Konstellationen entscheidende Vorteile haben können. Während einer langen Deployment-Phase sparen die einzelnen Knoten Energie, trotzdem wird das gesamte Netz innerhalb sinnvoller Zeit aufgeweckt und kann mit den eigentlichen Aufgaben beginnen. Durch diesen effizienten Umgang mit der kritischen Ressource Energie lässt sich die Lebensdauer eines solchen Sensorknotens entscheidend verlängern.
Zudem haben wir mit unserer Simulation empirisch gezeigt, dass der Probability Increase Algorithmus eine Verbesserung zum Birthday Algorithmus bringt. Trotz unserer grosszügigen Wahl der Konstante c, war er für sämtliche getesteten Dichten und Wahrscheinlichkeiten p_l schneller. Würde man sich auf kleine Werte für p_l beschränken, so würde sich sogar ein noch grösserer Vorteil ergeben.
Neben der Energieeffizienz sind für drahtlose Sensornetzwerke noch viele Fragenstellungen wie Skalierbarkeit, Robustheit, Mobilität oder Selbst-Konfiguration offen. So werden Sensornetzwerke in naher Zukunft ein spannendes Forschungsgebiet bleiben.

9 Referenzen

[**1**] Michael J. McGlynn und Steven A. Borbash. Birthday Protocols for Low Energy Deployment and Flexible Neighbor Discovery in Ad Hoc Wireless Networks.

[**2**] Tomasz Jurdzinski und Grzegorz Stachowiak. Probabilistic Algorithms for the Wakeup Problem in Single-Hop Radio Networks.

[**3**] Roger Wattenhofer und Aaron Zollinger. XTC: A Practical Topology Control Algorithm for Ad-Hoc Networks.

[**4**] L.G. Roberts. Aloha Packet System with and without Slots and Capture.

[**5**] Bluetooth project. www.bluetooth.com.