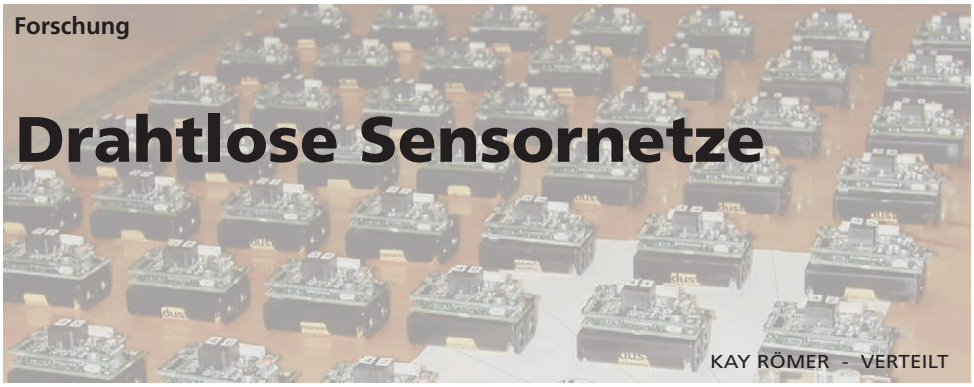


Forschung

Drahtlose Sensornetze

KAY RÖMER - VERTEILT

Drahtlose Sensornetze gestatten die detaillierte und weiträumige Beobachtung von Phänomenen der realen Welt, indem man eine Vielzahl so genannter Sensorknoten in die Umwelt ausbringt. Bei diesen Sensorknoten handelt es sich um Miniaturcomputer, die Umgebungsparameter über Sensoren erfassen, mittels Prozessoren verarbeiten und per Funk an benachbarte Sensorknoten übertragen können. Aufgrund der anvisierten Grösse auch manchmal als "Smart Dust" bezeichnet, sollen Sensorknoten im Batteriebetrieb über Monate oder Jahre hinweg ohne menschliches Zutun ihren Dienst verrichten.

1 Anwendungen

In einer Vielzahl von Anwendungsgebieten verspricht man sich einen grossen Nutzen von dieser Technologie, da man ohne aufwendige Installation (Kabel verlegen etc.) und Administration (Netzwerk konfigurieren etc.) diese Sensorknoten idealerweise einfach "ausstreuen" kann und neuartige Einblicke in die Vorgänge der realen Welt erhält. So können beispielsweise im Bereich der Gebäudeautomatisierung nachträglich und ohne Eingriffe in die Gebäudesubstanz Sensorknoten installiert werden, um eine Gesamtenergiebilanz zu ermitteln und zu optimieren. Man kann dafür Sensorknoten einfach an Wasserleitungen anbringen, um Menge (anhand des Durchflussgeräusches) und Temperatur des an bestimmten Stellen entnommenen Wassers zu ermitteln. Ebenso können Sensorknoten zur Messung der von einzelnen elektrischen Geräten entnommenen Energie (anhand des Magnetfelds) und zur Bestimmung von Temperatur und Qualität (z.B. Kohlendioxid-Gehalt) der Luft in Räumen verwendet werden.

Ein weiteres Anwendungsgebiet ist die Überwachung von Infrastruktureinrichtungen wie Pipelines oder Brücken. Beispielsweise könnten Sensorknoten an Ferngasleitungen angebracht werden, um mittels Ultraschallmessungen Korrosionsstellen zu entdecken und deren Ort zu melden. An Brücken können Sensorknoten zur Überwachung von windbedingten Schwingungen (mittels Beschleunigungssensoren) oder zur Entdeckung von Rissen in der Bausubstanz (mittels Ultraschallmessungen) installiert werden.

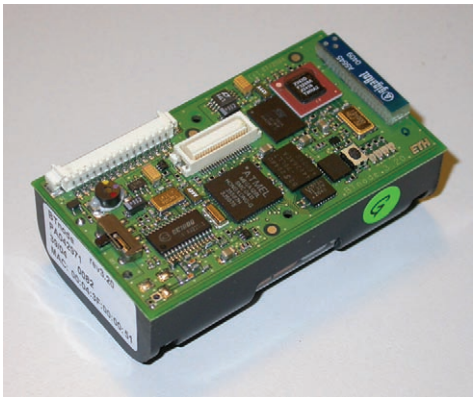
Auch für die wissenschaftliche Beobachtung von Naturerscheinungen können Sensornetze nutzbringend eingesetzt werden. So kann man zum Beispiel kleine Sensorknoten am Eingang der Bruthöhlen von Vögeln auslegen, um deren Brutverhalten ungestört zu beobachten (wann kommen und gehen die Vögel, wie viele sind jeweils im Nest etc.).

Je nach Anwendung ergeben sich hier eine Vielzahl verschiedener Anforderungen an die Hard- und Software von Sensorknoten. In vielen Fällen müssen die Sensornetze sehr langlebig sein (Monate oder Jahre) und daher sehr wenig Energie verbrauchen. Geringer Energieverbrauch impliziert wiederum, dass die Sensorknoten nur sehr wenig Rechenkapazität und Speicher für Anwendungssoftware bereitstellen können. Zudem sollten Sensornetze robust und ohne Eingreifen menschlicher Administratoren funktionieren - trotz einer Vielzahl an möglichen Störungen durch schädliche Umwelteinflüsse (Feuchtigkeit, hohe Temperaturschwankungen, mechanische Belastungen etc.).

2 Forschungsaktivitäten

Unsere Arbeitsgruppe “Verteilte Systeme” betreibt bereits seit einigen Jahren Forschung und Entwicklung im Bereich der drahtlosen Sensornetze. Im Folgenden möchte ich einen kurzen Einblick in unsere Aktivitäten auf diesem Gebiet geben.

Grundlage für praktische Experimente mit Sensornetzen ist die Verfügbarkeit eines Sensorknotens. Im Rahmen des vom Schweizerischen Nationalfonds geförderten MICS-Projektes [1] arbeiten wir zusammen mit dem ITET-Departement bereits seit ca. 5 Jahren an der Entwicklung solcher Sensorknoten, den so genannten BTnodes [2]. Die in



BTnode – ein Sensorknoten, der Bluetooth spricht.

der Abbildung gezeigte Version 3 dieser Hardware verfügt neben Bluetooth zur drahtlosen Kommunikation, Prozessor und Speicher über zahlreiche Anschlussmöglichkeiten für Sensoren. In unserer Arbeitsgruppe wird unter anderem ein Betriebssystem für diese Sensorknoten entwickelt.

2.1 Betriebssysteme für Sensorknoten

Da, wie oben bereits erwähnt, die Leistungsfähigkeit eines Sensorknotens im Vergleich zu typischen PCs sehr eingeschränkt ist, unterscheiden sich Betriebssysteme für Sensorknoten stark von typischen PC-Betriebssystemen wie Linux oder Windows. Während Letztere das dynamische Laden und nebenläufige Ausführen von Programmen per Multi-Tasking gestatten, führt der Prozessor eines Sensorknotens typischerweise genau eine Programmdatei aus, die zuvor (per Funk oder Kabel) in den dafür vorgesehenen speziellen Programmspeicher des Prozessors geladen wurde.

Diese einzige Programmdatei enthält nun sowohl die Funktionen des Betriebssystems als auch die

eigentliche Anwendung. Im Gegensatz zu etwa Linux oder Windows, wo das Betriebssystem durch Speicherschutzmechanismen des Prozessors weitgehend vor Fehlern in Anwendungsprogrammen geschützt ist, führt hier ein Programmierfehler in der Applikation oft zum Systemcrash. Beschränkte Speicher- und Prozessorkapazität führen auch dazu, dass die sonst übliche dynamische Speicherverwaltung (new, delete, ...) und Nebenläufigkeit (Threads) - wenn überhaupt - nur in sehr einfacher Form realisiert werden können.

Die Herausforderung bei der Entwicklung eines Betriebssystems liegt nun darin, trotz dieser widrigen Umstände eine einfach verwendbare, effiziente und robuste Systemumgebung für die Programmierung von Sensorknoten bereitzustellen. In dieser Hinsicht haben wir in den letzten Jahren zwei verschiedene Ansätze verfolgt. Version 1 des BTnode-Betriebssystems basiert auf der so genannten ereignisbasierten Programmierung. Dabei kann der Programmierer zu einer Menge von vorgegebenen Ereignissen wie “eine Nachricht wurde empfangen”, “eine gewisse Zeitspanne ist verstrichen” usw. Funktionen definieren, die immer dann abgearbeitet werden, wenn das zugehörige Ereignis auftritt. Man muss hier also eine Anwendung in viele kleine Programmstücke zerlegen, anstatt, wie sonst üblich, ein zusammenhängendes sequentielles Programm zu schreiben. Jeden, der einmal hardwarenah programmiert hat, wird das stark an Interrupt-Programmierung erinnern.

Diese Art der Programmierung ist zwar einerseits sehr effizient, andererseits aber umständlich und fehleranfällig in der Verwendung. In Version 2 des Betriebssystems, genannt BTnut, wird daher eine vereinfachte Form von Threads unterstützt. Dies geht zwar zu Lasten der Ressourcen (zum Beispiel benötigt jeder Thread einen eigenen Stack), ist aber oft wesentlich bequemer zu nutzen.

Trotz aller Anstrengungen hinsichtlich Benutzbarkeit des Betriebssystems muss man doch betonen, dass die erfolgreiche Programmierung eines Sensorknotens durchaus eine Herausforderung ist, bei der selbst erfahrene Experten zahlreiche Fehler machen können, die obendrein oftmals nur schwer zu finden sind. Anwendern ohne dieses Expertenwissen, die ein Sensornetz für praktische Aufgaben einsetzen möchten, ist diese Art der Programmierung wohl kaum zuzumuten. Unsere Arbeitsgruppe erforscht daher Konzepte und Werkzeuge, welche die Entwicklung und den Einsatz von Sensornetz-Anwendungen signifikant vereinfachen und diese viel versprechende Technologie damit erst Anwendern wie Biologen oder Seismologen zugänglich machen.

2.2 Globale Programmiermodelle

Bisher wurden Sensorknoten meist unter Verwendung der Programmiersprache C programmiert, wobei direkt auf Funktionen des Betriebssystem zugegriffen wird mit all den oben geschilderten Fallstricken. Anstatt wie bisher einzelne Knoten individuell zu programmieren, verfolgen wir den Ansatz, ein einziges globales Programm für das gesamte Sensornetz zu spezifizieren. Dieses globale Programm wird dann von einem speziellen Compiler in individuelle Programme für die einzelnen Sensorknoten übersetzt.

Darüber hinaus wird das globale Programm in einer speziellen deklarativen Programmiersprache formuliert, die vom Entwickler nur wenige Kenntnisse des zugrunde liegenden Systems erfordert. Anstatt sich in Fragen der Speicherverwaltung, des

Betriebssystems und Nebenläufigkeits-Aspekten zu verlieren, kann sich der Entwickler hier auf die eigentliche Anwendung und deren Funktionalität konzentrieren.

Dieses allgemeine Prinzip lässt sich nun auf verschiedene konkrete Probleme anwenden. Im Folgenden möchte ich ein Beispiel vorstellen, welches wir "Generische Rollenzuweisung" ("Generic Role Assignment") nennen [3]. Dem zugrunde liegt die Beobachtung, dass in vielen Anwendungen Sensorknoten in Abhängigkeit von ihren lokalen Eigenschaften (z.B. verbleibende Energie in der Batterie, Position im Raum) und gewisser Eigenschaften ihrer Nachbarknoten spezielle Funktionen - Rollen - im Netz übernehmen. Ein Beispiel hierfür ist das Clustering-Problem, wobei den Rollen MASTER und SLAVE Sensorknoten so zugewiesen werden sollen, dass jeder SLAVE-Knoten mindestens einen MASTER-Nachbar hat und MASTER-Knoten ausschliesslich SLAVE-Nachbarn besitzen. Dies wird häufig zur Datenverarbeitung verwendet, wobei SLAVE-Knoten ihre Sensor-Messdaten an einen benachbarten, oft leistungsfähigeren MASTER schicken, der die Daten von allen SLAVE-Nachbarn einsammelt, verarbeitet und zu einer Basisstation sendet.

Wir haben nun eine globale deklarative Programmiersprache entwickelt, mit der eine Vielzahl solcher Rollenzuweisungsprobleme spezifiziert werden kann. Ein Compiler übersetzt diese Spezifikationen in effiziente Programme, die auf den Sensorknoten ausgeführt werden und zu einer Zuweisung der Rollen gemäss Benutzervorgaben führen. Falls sich die Knoteneigenschaften ändern, Knoten ausfallen oder ihre Position verändern, werden die Rollenzuweisungen automatisch angepasst. Eine stark vereinfachte Variante des obigen Clustering-Problems sieht in dieser Programmiersprache wie folgt aus:

```
MASTER :: {
  count (1 hop) {
    role == MASTER
  } == 0
}
SLAVE :: else
```

was besagt, dass ein Knoten genau dann MASTER werden soll, wenn die Anzahl direkter Nachbarknoten ("count (1 hop)") mit Rolle MASTER ("role == MASTER") gleich Null ist. Andernfalls soll der Knoten die Rolle SLAVE annehmen.

Man sieht an diesem sehr einfachen Beispiel eindrücklich, dass man sich nicht mit Systemdetails wie Verschicken und Empfang von Nachrichten, Erkennen von Knotenausfällen usw. herumschlagen muss, sondern sich voll und ganz auf das eigentliche Problem der Rollenzuweisung konzentrieren kann.

Die Herausforderung bei dieser Forschungsarbeit ist, globale deklarative Programmiersprachen zu finden, die es erlauben, eine Vielzahl verschiedener Probleme kompakt und einfach zu beschreiben, die dann aber auch noch in effiziente Sensorknoten-Programme übersetzt werden können, welche mit den beschränkten Ressourcen der Sensorknoten auskommen. Es gilt hier, gute Kompromisse zu finden, denn je allgemeingültiger die Programmiersprache, desto schwieriger ist typischerweise die Übersetzung in effiziente Sensorknoten-Programme zu bewerkstelligen.

3 Weitere Aktivitäten

Neben den oben geschilderten und weiteren Forschungsarbeiten [4] sind wir in eine Reihe weiterer Aktivitäten im thematischen Umfeld der Sensornetze involviert. So organisierte unsere Gruppe kürzlich den European Workshop on Wireless Sensor Networks (EWSN 2006) [5], der vom 13. bis 15. Februar an der ETH Zürich stattfand. Dort haben Wissenschaftler aus aller Welt die neuesten Forschungsergebnisse zu Sensornetzen vorgestellt und praktisch demonstriert. Auf diesem Workshop wurden auch die Gewinner des Sentient Future Competition [6] vorgestellt, ein Wettbewerb, wo nach innovativen Anwendungen von Sensornetzen im Zeitraum der nächsten 10 Jahre gesucht wurde. Den zweiten Platz hat übrigens ein Informatik-Student der ETH gewonnen.

Für all diejenigen, die Gefallen an der Thematik der Sensornetze gefunden haben, werden wir im Sommersemester 2006 eine Vorlesung zum Thema anbieten. Dort wird neben grundlegenden und weiterführenden Aspekten der Sensornetze auch ein Praktikum angeboten, in dem die Teilnehmer selbst Anwendungen mit Sensornetzen realisieren können. Details dazu unter [7].

Referenzen

[1] www.mics.ch
[2] www.btnode.ethz.ch
[3] www.vs.inf.ethz.ch/publ/papers/sensys05.role-assignment.pdf
[4] www.vs.inf.ethz.ch/res/
[5] www.ewsn.org
[6] www.embedded-wisents.org/competition/competition.htm
[7] www.vs.inf.ethz.ch/edu/