TKN Telecommunication Networks Group

Technical University Berlin
Telecommunication Networks Group

1. GI/ITG KuVS Fachgespräch "Sensornetze"

H. Karl (Editor)

karl@ee.tu-berlin.de

Berlin, Juli 2003

TKN Technical Report TKN-03-012

TKN Technical Reports Series

Editor: Prof. Dr.-Ing. Adam Wolisz

Vorwort

Am 10. und 11. Juli 2003 fand das erste “GI/ITG Fachgespräch Sensornetze” statt. Es wurde gemeinsam durch das Leitungsgremium dieses Fachgespräches (Dr. Thomas Fuhrman, Universität Karlsruhe, Prof. KurtGeihs, Technische Universität Berlin, Dr. Holger Karl, Technische Universität Berlin, Prof. Friedemann Mattern, ETH Zürich, sowie Dr. Hartmut Ritter, FU Berlin) organisiert. 35 Teilnehmer aus Universitäten und Industrie waren angemeldet; es fanden 12 Präsentationen und die Vorführung einiger Demonstratoren statt. Dieser Bericht enthält die Beschreibungen dieser Präsentationen sowie einige der vorgestellten Foliensätze.

Die Organisatoren hoffen, daß diese Veranstaltung die Zusammenarbeit und Qualität der Forschung zu Sensornetzen verbessern wird.

Berlin, Juli 2003
Holger Karl

Tabelle 1. Teilnehmerliste

Name	Institution	email
Becker, Christian	U. Stuttgart	beckercn@informatik.uni-stuttgart.de
Beutel, Jan	ETH Zürich	j.beutel@ieee.org
Blumenthal, Jan	U. Rostock	jan.blumenthal@etechnik.uni-rostock.de
Buchmann, Alejandro	TU Darmstadt	buchmann@informatik.tu-darmstadt.de
Buschmann, Carsten	TU Braunschweig	buschmann@ibr.cs.tu-bs.de
Dalmases, Frances	Phillips	francesc.dalmases@phillips.com
Gober, Peter	FOKUS Fraunhofer	gober@fokus.fraunhofer.de
Golatowski, Frank	U. Rostock	Frank.Golatowski@etechnik.uni-rostock.de
Fuhrmann, Thomas	TU Karlsruhe	fuhrmann@tm.uka.de
Haase, Marc	U. Rostock	marc.haase@etechnik.uni-rostock.de
Hähner, Jörg	U. Stuttgart	haehnejg@informatik.uni-stuttgart.de
Handy, Matthias	U. Rostock	matthias.handy@etechnik.uni-rostock.de
Handziski, Vlado	TU Berlin	handzisk@ft.ee.tu-berlin.de
Herrmann, Klaus	TU Berlin	kh@ivs.tu-berlin.de
Hof, Hans-Joachim	TU Karlsruhe	hof@tm.uka.de
Huenerberg, Phillip	FOKUS Fraunhofer	huenerberg@fokus.fraunhofer.de
Hurler, Bernhard	TU Karlsruhe	hurler@tkm.uka.de
Kappler, Cornelia	Siemens AG	cornelia.kappler@siemens.com
Karl, Holger	TU Berlin	karl@ee.tu-berlin.de
Kobenstein, Jochen	U. Kiel	jko@informatik.uni-kiel.de
Köpke, Andreas	TU Berlin	koepke@ee.tu-berlin.de
Kubisch, Martin	TU Berlin	kubisch@ee.tu-berlin.de
Luttenberger, Norbert	U. Kiel	nl@informatik.uni-kiel.de
Marron, Pedro Jose	U. Stuttgart	marronpo@informatik.uni-stuttgart.de
Mattern, Friedemann	ETH Zürich	mattern@inf.ethz.ch
Mühl, Gero	TU Berlin	gmuehl@ivs.tu-berlin.de
Radusch, Ilja	FOKUS Fraunhofer	radusch@fokus.fraunhofer.de
Ritter, Hartmut	FU Berlin	hritter@inf.fu-berlin.de
Römer, Kay	ETH Zürich	roemer@inf.ethz.ch
Schiller, Jochen	FU Berlin	schiller@pcpool.mi.fu-berlin.de
Schmidt, Stefan	TU Braunschweig	schmidt@ibr.cs.tu-bs.de
Schmitz, Ralf	NEC	ralf.schmitz@ccrle.nec.de
Schwieger, Katja	TU Dresden	schwieg@ifn.et.tu-dresden.de
Sedov, Igor	U. Rostock	igor@informatik.uni-rostock.de
Todorova, Petia	FOKUS Fraunhofer	todorova@fokus.fraunhofer.de
Voigt, Thiemo	FU Berlin	voigt@pcpool.mi.fu-berlin.de
Willig, Andreas	Hasso-Plattner Institut, Potsdam	awillig@hpi.uni-potsdam.de

Inhaltsverzeichnis

Architektur

Testbeds, Erfahrungsberichte

“Lokale” Protokolle

Protokolle mit End-to-end Aspekten

SWARMS – Software Architecture for Radio-Based Mobile Self-Organizing Systems

Carsten Buschmann, Stefan Fischer

Institut für Betriebssysteme und Rechnerverbund
Technische Universität Braunschweig
Mühlenpfordtstr. 23, D-38106 Braunschweig
{buschmann|fischer}@ibr.cs.tu-bs.de

Jochen Koberstein, Norbert Luttenberger, Florian Reuter

Institut für Informatik und praktische Mathematik
Christan-Albrechts-Universität zu Kiel
Christian-Albrechts-Platz 4, D-24098 Kiel
{jko|nl|flr}@informatik.uni-kiel.de

1. Einleitung

Fortschritte hinsichtlich Miniaturisierung und Energieeffizienz von Hardware für Sensoren, Prozessoren und Speicher sowie Funkinterfaces haben mit den Sensornetzen eine neue Klasse von Netzwerken ermöglicht, die eine Fülle neuer Anwendungsfelder erschließen. So könnten beispielsweise über einem Waldbrandgebiet Tausende kleiner Temperaturfühler abgeworfen werden, die sich vernetzen und aufgrund der gemessenen Werte komplexere Informationen über den Brand wie etwa Position, Richtung und Geschwindigkeit von Feuerfronten liefern.

In dem beschriebenen Szenario finden sich Parallelen zu Schwärmen von z.B. Vögeln, Fischen oder Insekten, die mittels des Zusammenspiels von in ihren Fähigkeiten relativ beschränkten Individuen komplexe Aufgaben wie Nestbau oder Nahrungssuche bewältigen. In dem beschriebenen Netzwerk ergibt sich die Fähigkeit der einfachen Geräte, die Beantwortung komplexerer Fragestellungen zu unterstützen, ebenfalls durch Kooperation. Ihre große Zahl führt dabei zu neuen Herausforderungen hinsichtlich der Programmierung und Kooperation. Im SWARMS-Projekt wird untersucht, inwieweit die Programmierung von problemorientierten Anwendungen für gemeinsam operierende Schwärme von mobilen, funkvernetzten Systemen auf der Basis eines read/write Kooperationsparadigmas unterstützt werden kann [FL01]. In dieser Arbeit wird auf verschiedene Teilaspekte näher eingegangen: Zunächst wird der Begriff des Schwarmes näher erläutert. Im sich daran anschließenden Abschnitt wird das zugrunde liegende Datenmodell beschrieben. Der nächste Teil beschäftigt sich mit der geplanten Implementierung in einer Middleware. Der Beitrag wird durch die Betrachtung von Eigenschaften der avisierten Hardware abgerundet.

2. Schwärme

Das von uns betrachtete Kooperationsparadigma ist wie oben angedeutet durch das in der Tierwelt zu beobachtende Phänomen des Schwarms inspiriert. In Anlehnung daran wird der Begriff des Schwarmes hier wie folgt verstanden: Die *Einzelsysteme* verfügen über eine begrenzte Rechen- und Speicherleistung. Sie kommunizieren über ein infrastrukturloses Funknetzwerk, d.h. über ein geteiltes Medium. Dabei kann weder auf fehlerfreie noch auf ständige Konnektivität vertraut werden. Weiterhin muss davon ausgegangen werden, dass die meisten Einzelsysteme nicht über klassische Bedienelemente wie Tastatur oder Bildschirm verfügen; stattdessen sind sie über Sensoren oder Aktoren direkt in ihre Umwelt eingebettet. Hinsichtlich der Mobilität der Systeme lassen sich drei wesentliche Fälle unterscheiden. Im ersten Fall ist die weit überwiegende Anzahl der Geräte stationär (z.B. Sensorknoten). Im zweiten Fall sind die Geräte zwar mobil, die Applikation kann jedoch nicht über die Bewegung bestimmen (z.B. umher getragene PDAs). Der dritte Fall umfasst selbstbestimmte Bewegung, wofür autonome mobile Roboter ein Beispiel darstellen. Unabhängig von der

Mobilität muss damit gerechnet werden, dass eine optimale Platzierung der Systeme aufgrund von Anwendungsszenarien und Umweltgegebenheiten nicht möglich ist.

Die *Zusammensetzung* des Schwarmes ist variabel, die Veränderungen sind zumindest teilweise nicht im Voraus absehbar. Gründe für diese Veränderungen können menschliche Eingriffe sein, aber auch Mobilität, zeitweilige Kommunikationsabbrüche oder Gerätefehlfunktionen beispielsweise durch Batterieentleerung oder sogar Zerstörung. Somit sind die von einzelnen Systemen gelieferten Informationen weder persistent noch zuverlässig.

Der Schwarm verfolgt ein *definiertes gemeinsames Operationsziel*; dieses kann durchaus über die Zeit variieren. Es wird durch die auf alle Geräte einheitlich programmierte Anwendung vorgegeben und kann im Allgemeinen nur durch die Kooperation mehrerer Schwarmmitglieder erreicht werden. Dieses Ziel erfordert *situation awareness*: Die Geräte müssen sich des Kontextes bewusst sein, also z.B. ihrer Konfiguration, ihrer Umwelt oder ihrer eigenen Lokation. Letztere kann sowohl als absolute Position in Form von Koordinaten als auch kontextabhängig relativ (z. B. „am Ufer“) ausgedrückt werden. Dabei kann zur Ermittlung des Kontextes die Kooperation mit anderen Mitgliedern des Schwarmes nötig sein.

Aufgrund von Mobilität, sich wandelndem Kontext und funktionaler Beschränkung der Geräte ist eine feste Rollenzuweisung nicht sinnvoll. Daher haben die einzelnen Systeme *keine fest zugewiesenen Teilaufgaben*. Das Aushandeln der Aufgabenverteilung erfolgt selbstorganisiert und situationsabhängig. Beispielsweise könnte ein Gerät mit Sensoren und Festnetzanbindung je nach Bedarf die Funktionen Datenlieferant oder Festnetz-Gateway übernehmen.

3. Datenmodell

Die Lösung vieler Anwendungsprobleme setzt eine globale oder mindestens regionale Sicht auf die eigene Umgebung voraus. Nicht nur wegen der daraus resultierenden enormen Programmierkomplexität, sondern vor allem auch wegen des enormen Kommunikations- und damit verbundenen Energiebedarfs ist es nicht sinnvoll, diese Sicht durch den Austausch von direkt an einzelne Systeme gerichteten gerouteten Unicast-Nachrichten zu etablieren.

Daher fußt die Kooperation in SWARMS auf einem *Virtual Shared Information Space* (VSIS), in dem der Zustand sowohl der Umwelt als auch des Schwarmes selbst beschrieben ist. Es wird deshalb von einem „virtual“ shared information space gesprochen, weil es keinen Knoten im System gibt, der über alle Komponenten dieses Informationsraumes verfügt. Der VSIS setzt sich aus Informationen zusammen, die von den Schwarmmitgliedern akquiriert und verfügbar gemacht werden. Aus der Mobilität der Knoten, möglichen Geräteausfällen und der beschränkten Funkreichweite resultiert für jedes Mitglied eine unter Umständen inkonsistente Sicht auf einen Ausschnitt der Gesamtinformation.

Es gibt eine gewisse Ähnlichkeit zum Publisher/Subscriber-Modell [HG01], aber auch wesentliche Unterschiede: der VSIS kennt weder zentrale Instanzen zur Informationsverwaltung noch eine starre Rolleneinteilung in Informationsquellen und Senken.

Der VSIS wird aufgebaut aus semistrukturierten selbstbeschreibenden Informationskomponenten und durch ein XML-Schema in seinem Aufbau beschrieben. Diese Komponenten umfassen auch syntaktische und semantische Metainformationen zum Kontext der Informationsentstehung (räumlicher und zeitlicher Bezug, Verlässlichkeit) oder zu den

Daten selbst (Aggregationsgrad, Gültigkeitsbereich und -zeitraum). Diese helfen der Applikation, die Daten richtig zu nutzen.

4. Middleware

Um nun miteinander kommunizieren zu können, greifen Knoten mit Hilfe von Read-/Write-Operationen auf den VSIS zu. Die SWARMSware – eine im SWARMS-Projekt konzipierte Middleware – stellt eine Abstraktionsschicht dahingehend dar, dass der Programmierer eine Schwarm-Applikation als ein kohärentes Softwaresystem entwickeln kann, statt eine Menge unterschiedlicher Einzelsysteme zu programmieren.

Die SWARMSware verbirgt Selbstorganisation und Mobilität innerhalb eines Schwarms. Dazu gehören die Etablierung und Pflege von Ordnungen innerhalb des Schwarms (z.B. Regionen und Reihenfolgen). Der Aufwand zur Aufrechterhaltung solcher Ordnungen ist abhängig vom Grad der Mobilität der Schwarmmitglieder. Die SWARMSware stellt außerdem Mechanismen zur Etablierung eines gemeinsamen Zeitverständnisses zur Verfügung.

Das die VSIS-Struktur beschreibende XML-Schema legt fest, welche Struktur die Daten haben, die über die Zugriffsfunktionen der SWARMSware an die Anwendung übergeben werden können. Beim Zugriff der Applikation auf den VSIS erfolgt eine Transformation der Daten aus der XML-Darstellung in eine Repräsentation, die für die Applikation geeignet ist (Language Binding).

Beim Empfang von Informationen von anderen Knoten stellt die SWARMSware die Konformität der von anderen Knoten erhaltenen Daten zum XML-Schema sicher. Die Anwendung übergibt der SWARMSware Filter, die festlegen, welche dieser Informationen für sie von Interesse sind und in den VSIS-Ausschnitt aufgenommen werden sollen. Eine Herausforderung stellt die kompakte, die Anwendungsalgorithmen gut unterstützende und flexible Repräsentation der Daten dar. Die Applikation muss die Möglichkeit haben, sowohl die Anordnung als auch die Darstellung der Daten zu beeinflussen.

Eine weitere Aufgabe der Middleware ist die Unterstützung der *Kontextuierung und Bewertung von Informationskomponenten*. Metainformation wird z.B. durch die Applikation beim Schreibprozess oder durch die Middleware beim Erhalt von ähnlichen Daten (hinsichtlich Art, Zeit, Ort, etc.) erzeugt. Wenn ein Knoten z.B. weiß, dass mehrere direkte Nachbarn zu einem bestimmten Zeitpunkt eine Temperatur von 20°C messen, kann man aufgrund physikalischer Gesetze davon ausgehen, dass der von einem einzelnen Nachbarn übermittelte Messwert von 0°C nur bedingt vertrauenswürdig ist. Eine Möglichkeit zur Realisierung eines solchen Bewertungsmechanismus stellt das Voting mit Quoren dar, die von der Applikation spezifiziert werden. Dabei bleibt es der Anwendung überlassen, wie sie mit Informationen zur Verlässlichkeit von Daten umgeht. Modelliert werden solche Informationen entweder als XML-Attribute oder eigene XML-Dokumentenelemente.

Auf diesen Basismechanismen aufbauend unterstützt die Middleware zusätzlich die *Informationsaggregation*. Es werden Funktionen zur Transformation einer Menge von Informationselementen in eine kompaktere oder aussagekräftigere Darstellung zur Verfügung gestellt. Diese Transformationen sind anwendungsspezifisch und müssen nach von der Applikation vorgegebenen Regeln stattfinden.

Die Middleware soll auch die *Rollenverteilung* innerhalb des Schwarmes unterstützen. Die Aushandlung der Übernahme von verschiedenen Teilaufgaben durch einzelne Geräte erfolgt anhand des Kontextes und anhand der Fähigkeiten der Geräte. Der Kontext kann durch die Anwendung „gesetzt“ werden, oder nach im Vorfeld durch die Anwendung spezifizierten Kriterien von der Middleware ermittelt werden. Die Zuordnung von Aufgaben zu bestimmten Geräteeigenschaften und Kontextmerkmalen wird von der Anwendung nach ihren Bedürfnissen definiert.

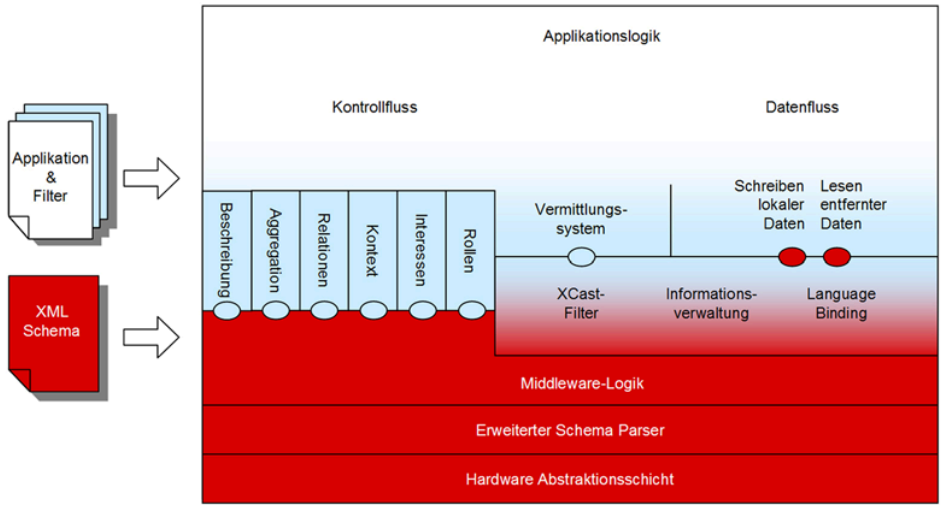


Abbildung 1: Architektur der SWARMSware mit der darauf aufsetzenden Anwendung

Im drahtlosen Netz eines Schwarms verbreiten die Knoten Informationen per adressenfreier Kommunikation (Broadcast). Knoten, die eine Information empfangen, leiten diese nur dann weiter, wenn sie für den Knoten selbst neu war bzw. mit eigener Information in geeigneter Art und Weise verschmolzen werden können. Ist die Menge der im Netz zu verteilenden Informationen endlich, lässt sich auf diese Weise die Anzahl der versendeten Pakete gegenüber simplem Fluten erheblich senken.

5. Hardware

Die von einigen Szenarien erforderte große Anzahl der benötigten Geräte sowie die große Verlustgefahr erfordern einen geringen Herstellungspreis. Darüber hinaus bedingen verschiedene Anwendungsszenarien eine starke Miniaturisierung [AAC+00], einige Forschungsgruppen legen ihren Überlegungen Geräte mit nur ein bis zwei Millimetern Kantenlänge zu Grunde [WLL+01]. Um die Batterielebensdauer der Geräte zu maximieren, ist ein geringer Stromverbrauch eine weitere Anforderung. Insgesamt muss davon ausgegangen werden, dass die in Sensornetzwerken eingesetzte Hardware eine sehr beschränkte Leistungsfähigkeit besitzt. So besitzen beispielsweise SensorMotes nur einen zwischen 8 und 128KB großen Befehlsspeicher sowie zwischen 512 Byte und 4 KByte Arbeitsspeicher [LC02]. Die Netzknoten kommunizieren über ein infrastrukturloses Funknetz; dabei handelt es sich um ein geteiltes Medium. Jeder Empfänger muss selbst entscheiden, ob eine ausgesendete Information für ihn interessant ist, es handelt sich um eine adressenlose Kommunikation. Somit findet auch kein Routing statt, es können höchstens Informationen in eine bestimmte Region geflutet werden. Da die Informationsübertragung mehr Energie benötigt als die Informationsverarbeitung [PG00], ist auf möglichst hohe Lokalität eingesetzter Verfahren zu achten.

6. Ausblick

Das SWARMS-Projekt befindet sich derzeit in der Anfangsphase. Neben einer groben Datenmodellierung wurden bisher Anforderungen und Ziele definiert, aber auch noch offene Forschungsfelder identifiziert. Die in dieser Arbeit beschriebenen Problemstellungen und Lösungsansätze sollen experimentell evaluiert und weiterentwickelt werden. Neben einer simulativen Erprobung soll auch mit einer heterogenen Handwarelandschaft und unterschiedlichen Kommunikationstechniken experimentiert werden. Dabei kommen PDAs (mit Bluetooth, WLAN, IrDA und RS-232-Schnittstelle), einfache mobile autonome Roboter (Kommunikation via IrDA) sowie Berkeley Mica Motes (mit einem 433MHz Funkinterface) zum Einsatz. Die verschiedenen Plattformen lassen sich miteinander kombinieren und bieten so eine breite, vielfältige Basis.

[AAC+00] Abelson, H., Allen, D., Coore, D., Hanson, C., Rauch, E., Sussmann, G.J., Weiss, R.: Amorphous Computing, Communications of the ACM, Vol. 43, No. 5, Mai 2000, 74-82

[FL01] Fischer, S., Luttenberger, N.: SWARMS – Software Architecture for Radio-Based Mobile Self-Organizing Systems, Projektantrag im Rahmen des DFG-Schwerpunktprogrammes „Basissoftware für selbstorganisierende Infrastrukturen für vernetzte mobile Systeme (SPP 1140), November 2001

[HG01] Huang, Y., Garcia-Molina, H. Publish/Subscribe in a Mobile Environment, International Workshop on Data Engineering for Wireless and Mobile Access, 2001, pp. 27-34

[LC02] Levis, P., Culler, D.: Maté: a Virtual Machine for Tiny Networked Sensors, ASPLOS, Dezember 2002.

[PG00] Pottie, G.J., Kaiser, W.J.: Wireless Integrated Network Sensors, Communications of the ACM, Vol. 43, No. 5, Mai 2000, 51-58

[WLL+01] Warneke, B., Last, M., Liebowitz, B., Pister, K.S.J.: Smart Dust: Communicating with a Cubic Millimeter Computer , IEEE Computer, 2001

SWARMS

Software Architecture for Radio-Based Mobile Systems

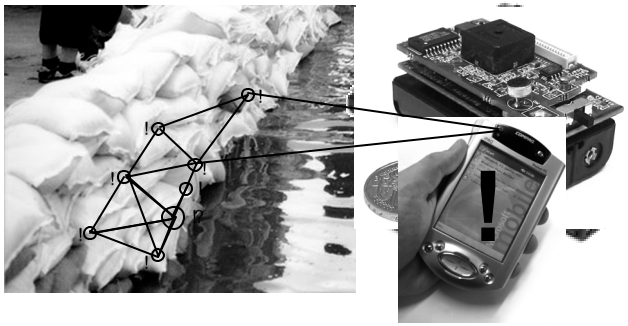
Carsten Buschmann, Stefan Fischer,
Jochen Koberstein, Norbert Luttenberger, Florian Reuter

10.7.2003, KuVS-Workshop Berlin

Ziele

- Erforschung von Middlewarekonzepten zur Unterstützung der Programmierung von selbstorganisierenden Schwärmen von mobilen Rechensystemen
  - Programmierung eines kohärenten Softwaresystems statt unterschiedlicher Einzelsysteme
  - Verbergen von Selbstorganisation und Mobilität des Schwarms gegenüber der eigentlichen Applikation

Szenario



Schwärme

- Definition aus dem Tierreich:
  - Verband mehrerer Einzelindividuen, die sich zusammenfinden, um gemeinsam verschiedene Zwecke zu erfüllen.
    - Schutz vor Feinden
    - Verbesserung des Jagderfolges
    - Aufzucht der Brut
- Schwarm mobiler Geräte:
  - Verband von Geräten
  - die ein gemeinsames Ziel verfolgen

Eigenschaften der Mitglieder

- Die Einzelsysteme
  - haben begrenzte Rechen- und Speicherfähigkeit,
  - werden oftmals über eine Batterie mit Energie versorgt,
  - kommunizieren über infrastrukturlose Funknetze weder mit totaler, noch mit dauerhafter Konnektivität,
  - sind in vielen Fällen über Sensoren und/oder Aktoren direkt in ihre Umwelt eingebettet (d.h. keine Ausstattung mit Tastatur, Maus, Bildschirm o.ä.),
  - sind unter Umständen voneinander unabhängig mobil.

Schwarmeigenschaften

- Zusammensetzung eines Schwarms nach Art und Anzahl der Einzelsysteme kann sich verändern
  - durch menschlichen Eingriff
  - durch die Bewegung und
  - durch Fehlfunktionen der Einzelsysteme
- gemeinsames Operationsziel des Schwarms kann nicht von einem Einzelsystem allein erreicht werden
- Context / situation awareness ist oft wichtig (wo bin ich, wer ist noch da ...)
- keine fest zugewiesenen Teilaufgaben für einzelne Mitglieder; die Aufgaben ergeben sich aus
  - Situation/Kontext und
  - Können der Geräte

Virtual Shared Information Space (VSIS)

- Problem erfordert häufig globale/regionale Sicht
- daher: Etablierung durch VSIS
  - beschreibt Zustand von Umwelt und Schwarm
  - Schwarmmitglieder sammeln Informationen und machen sie verfügbar
  - kein Knoten verfügt über alle Komponenten des Informationsraumes
  - jedes Mitglied hat ein lokale (und evtl. inkonsistente) Sicht auf einen Ausschnitt der Gesamtinformation
- Ähnlichkeit mit Publish/Subscribe Paradigma

Datenkomponenten

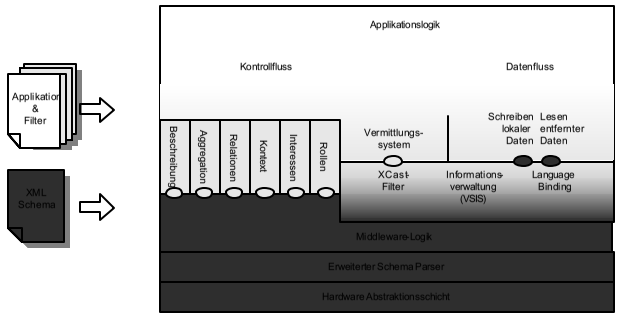
- VSIS besteht aus selbstbeschreibenden, semistrukturierten Informationskomponenten
- Augmentation mit syntaktischen und semantischen Metainformationen
  - Kontext der Informationsgewinnung (Ort, Zeit, Verlässlichkeit,…)
  - Metadaten (Aggregationsgrad, Gültigkeitsbereich/-zeitraum, …)
- Verwendung von XML/XML Schema

Aufgaben der Middleware

- Realisiert den VSIS
  - Zugriffsfunktionen
  - Informationsweiterleitung
- Verbergen der Verteiltheit
  - Selbstorganisation (gemeinsames Verständnis von Ort und Zeit)
  - Mobilitätstransparenz
- Unterstützung von
  - Kontextuierung und Bewertung
  - Informationsaggregation
  - Rollenverteilung

AG Kommunikationssysteme
Inst. für Informatik u. Prakt. Mathematik
Christian-Albrechts-Universität zu Kiel
Inst. für Rechnerverbund und Betriebssysteme
TU Braunschweig

Architektur



AG Kommunikationssysteme
Inst. für Informatik u. Prakt. Mathematik
Christian-Albrechts-Universität zu Kiel
Inst. für Rechnerverbund und Betriebssysteme
TU Braunschweig

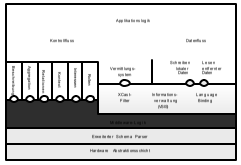
Realisierung des VSIS

- Zugriffsfunktionen für die Applikation
  - read und write auf den lokalen Ausschnitt des VSIS
  - XML-Schema beschreibt die Komponenten und somit die Schnittstellen
- Vermittlungssystem
  - Informationsverteilung zwischen den einzelnen VSIS-Ausschnitten
  - XCast content based forwarding: Mitglied leitet nur Daten per Broadcast weiter, die ihm selbst neu waren
  - Jeder Empfänger entscheidet, ob die Informationen für ihn relevant sind

AG Kommunikationssysteme
Inst. für Informatik u. Prakt. Mathematik
Christian-Albrechts-Universität zu Kiel
Inst. für Rechnerverbund und Betriebssysteme
TU Braunschweig

Selbstorganisation

- Verbergen der Verteiltheit durch Selbstorganisation
  - Erkennung von Nachbarschaftsverhältnissen
  - Automatischer Aufbau und Erhalt von
    - relativem oder absolutem Raumverständnis
    - gemeinsamem Zeitverständnis
    - Regionen, Gruppen und Reihenfolgen



AG Kommunikationssysteme
Inst. für Informatik u. Prakt. Mathematik
Christian-Albrechts-Universität zu Kiel
Inst. für Rechnerverbund und Betriebssysteme
TU Braunschweig

Applikationsfilter

Applikation stellt Funktionalität bereit für

- Datenselektion (Interessen)
- Kontextuierung/Bewertung (z.B. Voting)
- Informationsaggregation (Transformation zu kompakteren Ereignissen)
- Kontextdefinition
- Rollenverteilung (Kontextabhängig)
- …

AG Kommunikationssysteme
Inst. für Informatik u. Prakt. Mathematik
Christian-Albrechts-Universität zu Kiel
Inst. für Rechnerverbund und Betriebssysteme
TU Braunschweig

Hardware

Anforderungen

- geringer Preis
- Miniaturisierung
- Energieeffizienz

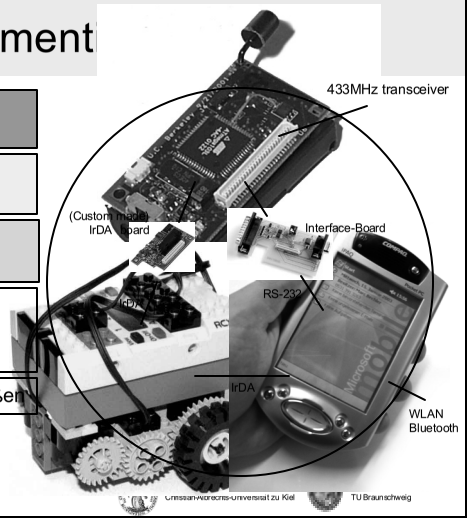
Resultat:

- beschränkte Hardware
- geteiltes Broadcast-Funkmedium mit beschränkter Reichweite

AG Kommunikationssysteme
Inst. für Informatik u. Prakt. Mathematik
Christian-Albrechts-Universität zu Kiel
Inst. für Rechnerverbund und Betriebssysteme
TU Braunschweig

Experiment

- iPAQ PDA, WLAN
- Savaje
- Lego RCX
- LegOS, Lejos
- Berkeley Motes
- TinyOS
- Heterogene IT
- Unterschiedliche Kommunikation
- Anbindung nach außen



Christian-Albrechts-Universität zu Kiel
TU Braunschweig

Vielen Dank.

AG Kommunikationssysteme
Inst. für Informatik u. Prakt. Mathematik
Christian-Albrechts-Universität zu Kiel
Inst. für Rechnerverbund und Betriebssysteme
TU Braunschweig

A common wireless sensor network architecture?

Vlado Handziski, Andreas Köpke, Holger Karl, Adam Wolisz
Telecommunication Networks Group
Technische Universität Berlin
handzisk,koepke,karl,wolisz@ft.ee.tu-berlin.de

1 Fragmentation of research

Wireless sensor network (WSN) research is currently heavily fragmented: Independent developments take place on both sides of the Atlantic, in many different research groups. There is not yet a common *"lingua franca"* as in some other fields of networking research, where e.g. Linux on Intel processors or the Network Simulator [1] have achieved such a status—with all the advantages and disadvantages of such a predominance. In WSNs, the need for a harmonization of research efforts is particularly felt in the following areas.

Simulation environments Many different research groups use different simulation environments for performance evaluation. Currently popular tools include NS/2, Opnet, GlomoSim, Qualnet, or OmNET++. None of these simulation tools perfectly meet the demands of WSNs. For example, the wireless channel model is too simplistic and not easily changed (NS/2), not all relevant protocols are easily available (Omnet), some are commercial (Opnet, Qualnet), some are too infrequently used to easily allow comparison of results with work of other researchers (all tools with the possible exception of NS/2).

More importantly, there is no simulation tool for which convincing models of applications or sensor excitation are available.

Hardware platforms The overall setup of sensor node prototypes that have been developed and are currently used by different groups for experimental research is in principle very similar. Examples for such nodes include the Mica Motes, the nodes used by the EYES project [2], and nodes developed locally at Technische Universitt Karlsruhe, ETH Zrich, Freie Universitt Berlin, or Technische Universitt Dresden. Typical microcontrollers are the Atmel, Motorola microcontrollers or the Texas Instruments MSP 430; usable radio modems include those by RFM, Chipcon, Maxim, Nordic, RFMD, Xemics, Infineon and others; also, Bluetooth chipsets are sometimes used.

While all of these nodes have their individual advantages and disadvantages and the level of diversity is natural for the current state of WSN research, a lot of synergy could be gained if a convergence to a single prototype could be achieved. The economies of scales alone would make such an effort worthwhile by making prototype nodes available in larger quantities at low cost. The Mica Motes offerings by Crossbow are a step in this direction, but are still somewhat expensive. Possibly, the current developments in the IEEE 802.15.4 standardization process will create sufficient pressure towards unification.

Moreover, this multitude of prototypes often renders experimental results difficult to compare. Radio modems have different modes of operations with different power consumptions, resulting in differently optimized lower-layer protocols. While these solutions are important and necessary, what is still missing is the next logical step in comparing the design decisions for example for MAC layer protocols that result from various properties of a specific node.

Operating systems Different prototypes are beneficial as long as their idiosyncracies can be abstracted away if necessary. Usually, an operating system performs this role of hardware abstraction. The design decisions taken for various prototypes vary quite a bit: from a simple hardware abstraction over an run-time environment like TinyOS to a full-blown task scheduler like in the EYES OS.

Again, the problem of comparability arises: How can own solutions be compared with established solutions and protocols that have been developed for a given operating system?

Development environments Sometimes, the hardware abstraction/operating system mandates the use of given development environments like MS Windows or commercial compilers like IAR. Typically, publicly available tool chains are preferable.

2 Three approaches towards better integration

One important step for overcoming these overcomings would be a harmonized simulation and experimental environment, which is a somewhat utopian goal. Nonetheless, the following three issues could be attempted.

Hardware abstraction for new sensor node prototypes While a complete porting of an arbitrary operating system to a new hardware is a painful undertaking, it might be reasonable to look for a useful hardware abstraction set that could be supported by most or all of the existing and imagined sensor nodes, that would be expandable to integrate future possibilities, and on top of which operating systems or runtime systems could be easily ported or developed.

The TinyOS abstraction layer is an excellent start in this direction; also, some work done at Freie Universitt Berlin could be leveraged here.

The advantage for hardware developer is the wider range of possible environments which could use a new hardware; the advantage to software developers is the wider range of usable hardware.

Ideally, for each sensor node prototype in practical use, such a hardware abstraction layer should be made publicly available.

Abstract model of power consumption As different, heterogenous hardware is here to stay, still a fairly accurate and *standardized* model of the capabilities of hardware is required, with respect to mainly power/energy consumption and performance. Both the radio front end and the microcontroller of a sensor node should be included in this model. If the actual sensor or actuator hardware also noticeably contributes to power consumption (as is the case e.g. for magnetometer sensors), it should also be mentioned.

The most relevant aspects of such an energy/power consumption model are:

- How many modes of operation can be distinguished for the microcontroller (sleep modes, operation, potentially reduced clock rate, ...) and the radio front end (sleep, idle, receive, transmit, ...)? Are there any other functional parts for which relevant state transitions occur, and what are they?
- What is the power consumption for these blocks in each of these modes? Does it depend (in which way) on additional parameters, e.g., the modulation?
- In case transmission power control is available: What is the mapping from radiated power to consumed power? Does it depend on additional parameters, e.g., modulation?
- What is the transition time from each operational mode to each other one (if the transition is directly possible at all), what is the power consumed during such a state transition (and, hence, the energy)?

- What are the performance parameters of the radio modem, especially, the receiver sensitivity, the maximum output power, the blocking characteristics, the mapping from SINR to bit error rate (depending on data rate and modulation), the available data rate(s), number of available channels?
- What are additional quality parameters like temperature drift of the radio front end or frequency stability, calibration of received signal strength indication (RSSI) signals, etc.?
- Any additional "unusual" characteristics of the microcontroller or the radio modem, e.g., the possibility to emit a busy tone while receiving data? Is the level of forward error correction or channel coding controllable, etc.?

While most of this information is available in some form or another in manufacturer's data sheets, a single, easy to access repository of this type of information is missing. Such a repository should be quite valuable, especially when evaluating communication protocols by simulating, providing a reference to actually realistic assumptions about the behavior of a range of real hardware. Moreover, experience reports like "often bad frequency stability" would add to the usefulness.

A Protocol Architecture Scheme for WSN One reason that makes the implementation of wireless networks in general and wireless sensor networks in particular so challenging is the need to exchange information between functionalities that belonged, in the traditional ISO/OSI model, to different protocol layers. A simple example is link-layer triggering for handover in cellular networks, transmission-power-aware routing protocols in ad hoc networks, and the multitude of mutual influences of protocol functionalities in wireless sensor networks.

In such a situation, the reuse of communication protocols, especially from a thrid party, is challenging. Ideally, individual "building blocks" for wireless sensor network protocols should be identified and implemented separately, with different ideas for the mechanisms realizing the desired functionalities. These building blocks would pass packets between each other, but also exchange meta-information, e.g., concerning the topology of the network or about the geographic position of individual nodes. In this sense, there will be building blocks that are predominantly engaged in packet exchange (e.g., a link layer building block) while others will be mostly tasked with the collection and computation of meta information (e.g., a building block that provides estimates about a sensor node's position).

When trying to structure a WSN protocol suite along these lines, identifying the most relevant building blocks and their most important packet passing and information exchange relationships, a structure like the one in Figure 1 is a possible outcome (depicting an intermediate state of the EYES project's architecture discussion).

This approach results in two challenges: How to organize the passing of a packet to the correct next building block; the second challenge is how to then organize the information exchange between building blocks. The first challenge is akin to the same problem in conventional protocol stacks with well-known solutions like packetfilters [3] which should also be applicable here. The second one is more difficult since it is neither clear *when* information is required from another building block nor *to whom* to deliver information. As an example, a building block for location estimates might be used by additional building blocks even after this block has been completed.

Hence, a separation of communication in time and space is required—the publish/subscribe model is doing exactly this. Providers of information publish it under a given "name", users of information can subscribe to such names and be informed of any value changes. One example for such a name would be RSSI: the physical layer could publish this information, and any layer that is interested in it could subscribe to it. One practical implementation concept for such a publish/subscribe structure is a "blackboard", where each building block can "write down" values for a given name and where a building block can post a callback for a given name to be invoked when a value for this name is initialized or changes.

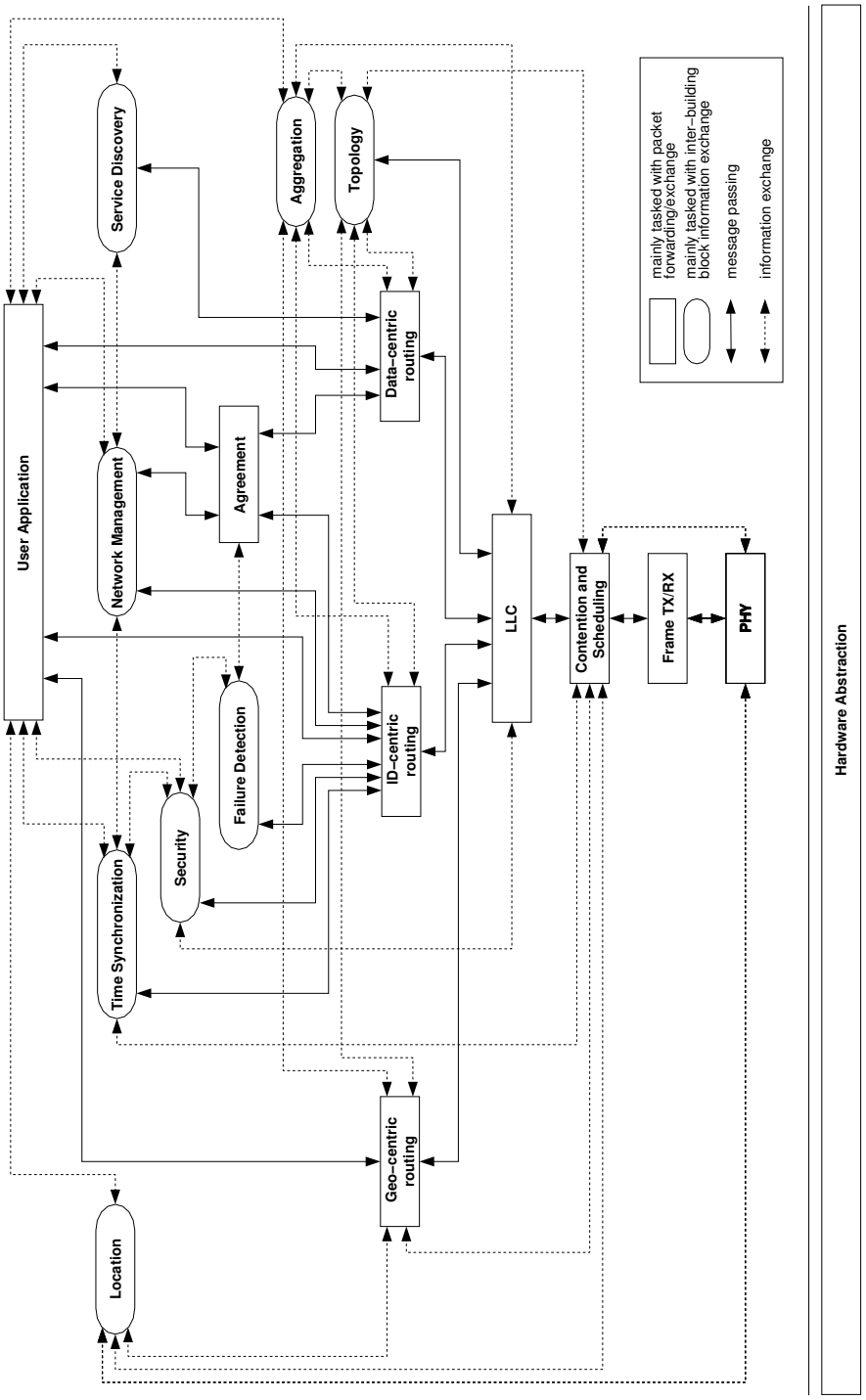


Figure 1: Possible structure of a WSN protocol suite

The advantage of such a publish/subscribe approach is the separation of individual building blocks, making the individual, separate development of protocol mechanisms easier.

Ideally, a protocol architecture should therefore be feasible that specifies names for such published information along with an appropriate packetfilter interface. In addition, some configuration language is necessary to describe which type of information a given building block relies upon and which therefore has to be present. In this way, a simple mixing and matching of individual building blocks from arbitrary sources and a comparison of different implementations of the same building block using different mechanisms should become easier.

3 Conclusion

The fragmentation of current WSN research is difficult to overcome and a complete harmonization of simulation and experimental environments is probably illusionary. Nevertheless, this paper has identified three approaches which could contribute to a better comparison and reuse of existing solutions: hardware abstractions for prototypes, characterizing the power consumption of actual hardware, and one possible approach to structure WSN protocol architectures that enables reusability in the face of the necessary tight integration of protocols in WSN.

Acknowledgements

Some of the content of this paper has been influenced by discussions with our colleagues from the EYES project, specifically, Thomas Lentsch, Michele Zorzi, and Paul Havinga; colleagues from FU Berlin, specifically Jochen Schiller and Hartmut Ritter; Adam Wolisz; and Andreas Willig. This work has in part been sponsored by the IST EYES project.

References

[1] The Network Simulator — ns-2. Project homepage at `http://www.isi.edu/nsnam/ns/`.

[2] EYES Project Website. `http://www.eyes.eu.org`, January 2003. Date of access.

[3] J. C. Mogul, R. F. Rashid, and M. J. Accetta. The Packet Filter: An Efficient Mechanism for User-level Network Code. In *Proc. 11th ACM Symp. on Operating System Principls*, Austin, TX, November 1987.

A common wireless sensor network architecture?

Vlado Handziski, Andreas Köpke,
Holger Karl, Adam Wolisz

Telecommunication Networks Group
Technische Universität Berlin

Fragmentation of research

- Many different simulation environments
- Different hardware platforms
- Different operating systems
- Development environments

➔Research results are not comparable

➔(Almost) everybody is building (almost) everything from scratch

Hardware abstraction for operating systems

- Porting complete operating systems to new hardware is time-consuming and bears no immediate return on investment
- Converging on a single hardware platform is unlikely to happen and not necessarily a good thing

➔A common hardware abstraction layer on top of which operating systems can be easily implemented?

- For any type of sensor node that is publicly available
- TinyOS abstraction layer, some work at FU Berlin

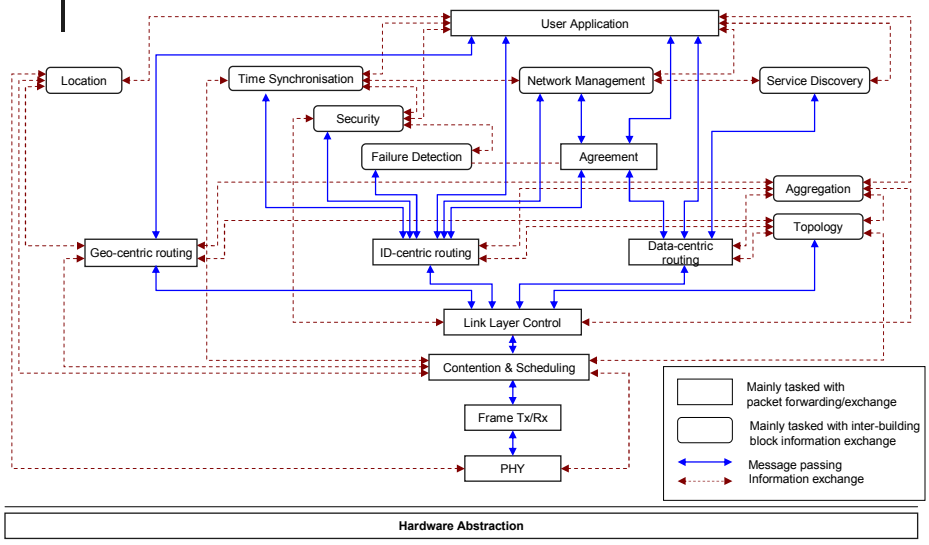
Abstract model of power consumption

- Power consumption of sensor nodes is their key characteristic
  - Common set of parameters?
  - Should be available for all nodes
- Typical parameters
  - Relevant blocks: Radio front end, microcontroller
  - Number of modes of operation for each block
  - Power consumption in each mode
  - Turn-around times between these modes
  - Transmission power mapped to consumed power
    - Additional factors, like modulation?
  - Performance characteristics of radio front end
  - Quality parameters: temperature drift, calibration, …?
  - "Unusual" characteristics: dual-channel radio, etc.

A protocol architecture scheme

- Strict ISO/OSI layering is hardly appropriate for WSNs
- Strong interaction between individual protocol building blocks
- How to handle such interaction when blocks come from different sources?
- One approach:
  - Publish/subscribe for cross-layer communication
  - "Configuration language" how to connect building blocks

Example architecture



Sicherheit in Sensornetzen am Beispiel von SWARMS

Stefan Schmidt, Carsten Buschmann und Stefan Fischer
TU Braunschweig, Institut für Betriebssysteme und Rechnerverbund
Mühlenpfordtstr. 23, D-38106 Braunschweig
{schmidt|buschmann|fischer}@ibr.cs.tu-bs.de

Viele Forschungsarbeiten über Sicherheit in Sensornetzen gehen von recht unterschiedlichen Prämissen aus. Für generischere Ansätze muss man auf spezifische Forderungen wie Geräte mit besonderen Fähigkeiten (z.B. leistungsfähige Prozessoren) oder Basisstationen verzichten und auf gleichwertige Geräte setzen, die dynamisch unterschiedliche Aufgaben übernehmen können, wobei jedwede Kommunikation zwischen den Knoten drahtlos geschieht. Die Vermeidung komplexer und zerbrechlicher logischer Hierarchien wie netzweite Baumstrukturen oder zentrale Instanzen verbessert Skalierbarkeit und Robustheit. Um die Kosten pro Gerät niedrig zu halten, muss momentan auch auf physikalisch manipulationssichere Geräte verzichtet werden.

In dieses Umfeld ist auch das SWARMS-Projekt einzuordnen. In diesem wird untersucht, inwieweit die Programmierung von problemorientierten Anwendungen für gemeinsam operierende Schwärme von mobilen, funkvernetzten Sensorknoten auf der Basis eines Read/Write-Kooperationsparadigmas unterstützt werden kann [2]. Die Lösung vieler Anwendungsprobleme setzt eine globale oder mindestens regionale Sicht auf die eigene Umgebung voraus. Daher fußt die Kooperation der Knoten in SWARMS auf einem *Virtual Shared Information Space* (VSIS), in dem der Zustand der Umwelt und des Schwarmes selbst beschrieben ist und der in einer Middleware implementiert wird. Dabei verfügt kein System über die gesamte Information des Schwarmes, vielmehr repräsentiert jedes Gerät einen Teilausschnitt des VSIS. Über Mechanismen zur Informationsverbreitung mittels adressenloser Kommunikation (Broadcast) sowie zum Verbergen der Verteiltheit durch Selbstorganisation hinaus soll die Middleware die transparente Programmierung verteilter Anwendungen durch semiautomatische Bewertung und Kontextuierung von Informationen, Informationsaggregation sowie dynamische Aufgabenverteilung unterstützen.

Sensornetze kommen im Allgemeinen in einem unkontrollierbaren, nicht vertrauenswürdigen Umfeld zum Einsatz, woraus sich ein erhöhter Bedarf an Sicherheit ableitet. Hierfür können die gebräuchlichen Sicherheitsprimitive Authentifizierung, Integrität, Vertraulichkeit und Verfügbarkeit auch in Sensornetzen angewendet werden. Ihr Einsatz ist generell auf das Absichern des Netzes gegenüber dem Zugriff Dritter ausgerichtet. Unerlässlich sind Sicherheitsmaßnahmen beispielsweise zum verlässlichen Austausch von Daten, der sicheren Datenaggregation innerhalb des Sensornetzes oder beim Einsatz von mobilem Code. Generell muss dabei beachtet werden, dass durch Sicherheitsmaßnahmen aufgrund der Berechnung immer Latenzzeiten und Energieverbrauch erhöht und eventuell auch zusätzliche Nachrichten zur Koordination der Sicherheitsmaßnahmen im Netz benötigt werden.

Im Bereich mobiler Ad-hoc-Netze werden zahlreiche vielversprechende Sicherheitsansätze diskutiert, die unter anderem die Risiken der inherent unsicheren, drahtlosen Kommunikation und der Probleme, die durch das Fehlen einer Infrastruktur entstehen, betrachten. Grundsätzlich sind in Sensornetzen die gleichen Problemstellungen wie in Ad-hoc-Netzen zu berücksichtigen. Darüber hinaus müssen jedoch weitere wichtige Randbedingungen berücksichtigt werden, was Sicherheitslösungen aus Ad-hoc-Netzen nur bedingt auf Sensornetze übertragbar macht. Vor allem die extrem geringe Leistungsfähigkeit der Sensorknoten hinsichtlich vorhandenem Speicher, Rechenleistung und Energie schließt etliche Sicherheitslösungen aus. So sind zum Beispiel Verfahren, die auf asymmetrischer Kryptographie beruhen, in vielen Sensornetzen nicht nutzbar, da sie hohe Ansprüche an Rechenleistung und Speicherplatz stellt. Weiterhin liegen Sensornetzen dynamische Topologien mit potenziell extrem vielen Knoten zu Grunde. Eine Sicherheitslösung muss somit extrem skalierbar und möglichst selbstadministrierend sein, da beispielsweise das manuelle Einführen eines neuen kryptographischen Schlüssels bei tausenden Sensorknoten nicht effizient gesteuert werden kann. Ebenfalls aus Gründen der Skalierbarkeit und um einen möglichst gleichmäßigen Energieverbrauch der einzelnen Knoten zu erreichen, sollten außerdem Berechnungs- und Verkehrskonzentrationen im Netz vermieden werden. Da weiterhin Messwerte und Ereignisse abhängig von der Anwendung schnell veralten können, müssen sie zeitnah kommuniziert werden, was Sicherheitsmaßnahmen mit langen Berechnungszeiten ausschließt. Schließlich müssen noch Datenaggregation innerhalb des Netzes und der Einsatz mobilen Codes beim Entwickeln einer Sicherheitslösung für Sensornet-

ze berücksichtigt werden.

Obwohl bereits einige Sicherheitsansätze speziell für Sensornetze existieren, berücksichtigt unserer Kenntnis nach kein Ansatz alle der oben genannten Randbedingungen, bzw. es werden Vereinbarungen getroffen, die nicht mit dem SWARMS-Konzept harmonieren. So benötigen manche Ansätze zwingend eine Basisstation, die als zentrale Instanz im Netz sicherheitsrelevante Aufgaben übernimmt [1, 3, 4, 5]. Weiterhin werden teilweise logisch strukturierte Netze gefordert, die dynamisch Hierarchien im Netz bilden [1, 3], oder physikalisch manipulationssichere Geräte [1].

Folglich ist für SWARMS ein generischeres Sicherheitskonzept erforderlich, welches die gegebenen Randbedingungen berücksichtigt. Grundsätzlich müssen immer die spezifischen Einschränkungen von Ad-hoc-Netzen (unsichere Kommunikation, fehlende Infrastruktur, etc.), Sensornetzen (Skalierbarkeit, Echtzeit-Anforderung, etc.) und der einzelnen Knoten (eingeschränkter Speicher, Rechenleistung, Energie, etc.) berücksichtigt werden. Darüber hinaus ist es notwendig, ein flexibles Konzept zu entwickeln, in dem sich der Grad der Sicherheit bei Bedarf variieren lässt. Dies ermöglicht insbesondere einen der Anwendung angepassten Kompromiss zwischen benötigter Sicherheit und dem resultierenden Energieverbrauch. In diesem Zusammenhang differenzieren wir hinsichtlich der beiden Dimensionen *inhaltsbasierte Sicherheit* und *situationsbasierte Sicherheit*. Inhaltsbasierte Sicherheit gründet auf der Prämisse, dass unterschiedliche Inhalte auch unterschiedliche Sicherheitsanforderungen haben. Zum Beispiel hat mobiler Code üblicherweise viel höhere Sicherheitsanforderungen als kommunizierte Messwerte oder Lokationsinformationen. Dies ermöglicht es der Middleware, für unterschiedliche zu kommunizierende Inhalte flexibel verschiedene Sicherheitsgrade festzulegen. Situationsbedingte Sicherheit beschreibt unterschiedliche Sicherheitsanforderungen abhängig vom internen Zustand des Sensornetzes und der Umwelt. Zwei Situationen, in denen für gleiche zu kommunizierende Daten (z.B. Lokationsinformationen der Knoten) verschiedene Grade an Sicherheit wünschenswert sind, sind zum Beispiel die Langzeitüberwachung von Deichen oder die Überwachung der Planung eines terroristischen Anschlages. Hierbei kann die Anwendung den gewünschten Grad der Sicherheit sowohl für anwendungsrelevante Daten, wie auch für Daten, welche die Middleware zur internen Organisation benötigt, festlegen.

Sicherheit in Sensornetzen kommt anwendungsabhängig eine unterschiedlich starke Bedeutung zu. Aus diesem Grund sind wir davon überzeugt, dass aufbauend auf den Basisanforderungen, die immer beachtet werden müssen, ein flexibles Sicherheitskonzept entwickelt werden muss. Denn so kann der optimale Kompromiss zwischen dem benötigten Grad an Sicherheit und dem Energieverbrauch erreicht werden. Zusätzlich sollte dieser Ansatz möglichst generisch sein, um die Anforderungen an ein Sensornetz, bzw. der einzelnen Knoten möglichst gering zu halten.
In dem Vortrag wollen wir zunächst allgemein das Problemfeld der Sicherheit in Sensornetzen beschreiben und Voraussetzungen und Limitationen aufzeigen. Im Hauptteil werden wir dann auf die beiden Paradigmen inhalts- und situationsbedingter Sicherheit eingehen und erste Ideen zu ihrer Umsetzung vorstellen. Dabei soll es sich nicht um die Präsentation fertiger Konzepte handeln. Vielmehr wollen wir erste konzeptionelle Ansätz beschreiben, die im weiteren Projektverlauf simulativ und experimentell evaluiert und weiterentwickelt werden sollen. Auf diese Art und Weise soll eine Diskussion ausgelöst werden, da wir von der Wichtigkeit des Themas überzeugt sind.

[1] Basagni, S. et al., *„Secure Pebblenets"*, MobiHoc 2001.

[2] Fischer, S., Luttenberger, N., *„SWARMS – Software Architecture for Radio-Based Mobile Self-Organizing Systems"*, Projektantrag im Rahmen des DFG-Schwerpunktprogrammes „Basissoftware für selbstorganisierende Infrastrukturen für vernetzte mobile Systeme" (SPP 1140), 2001.

[3] Hu, L., Evans, D., *Secure Aggregation for Wireless Networks*, Workshop on Security and Assurance in Ad-hoc-Networks, 2003.

[4] Perrig, A. et al., *„SPINS: Security Protocols for Sensor Networks"*, MobiCom 2001.

[5] Undercoffer, J. et al., *„Security for Sensor Networks"*, CADIP Research Symposium 2002.

Sicherheit in Sensornetzen

am Beispiel von SWARMS

Stefan Schmidt
10.07.2003

TECHNISCHE UNIVERSITÄT
CAROLO-WILHELMINA
ZU BRAUNSCHWEIG
Institut für Betriebssysteme und Rechnerverbund
http://www.ibr.cs.tu-bs.de

Überblick

- Problemfeld und Ziel der Arbeit
- SWARMS
- Problematiken in Sensornetzen
- Lösungsansatz
- Weiteres Vorgehen

TECHNISCHE UNIVERSITÄT
CAROLO-WILHELMINA
ZU BRAUNSCHWEIG
Institut für Betriebssysteme und Rechnerverbund
http://www.ibr.cs.tu-bs.de

Problemfeld

- Einsatzgebiete von Sensornetzen meist unkontrollierbar und nicht vertrauenswürdig.
- Absichern des Netzes gegen Zugriff Dritter
  - verlässlicher Austausch von Daten
  - Einsatz von mobilem Code

→ Schaffen eines möglichst generischen Sicherheitskonzept, das nur geringe Anforderungen an Netzkomponenten stellt.

TECHNISCHE UNIVERSITÄT
CAROLO-WILHELMINA
ZU BRAUNSCHWEIG
Institut für Betriebssysteme und Rechnerverbund
http://www.ibr.cs.tu-bs.de

SWARMS

Wesentliche Charakteristika:

- Kooperation gleichwertiger Geräte
- keine logische Netzstruktur
- dynamische Verteilung und Bewältigung von Aufgaben
- Knoten nicht physikalisch manipulationssicher

TECHNISCHE UNIVERSITÄT
CAROLO-WILHELMINA
ZU BRAUNSCHWEIG
Institut für Betriebssysteme und Rechnerverbund
http://www.ibr.cs.tu-bs.de

Probleme

- Ad-hoc-Netz Problematiken
  - fehlende Infrastruktur, etc.
- Sensornetz Problematiken
  - Skalierbarkeit, Echtzeitanforderungen, etc.
- Sensorknoten Problematiken
  - geringe Rechenleistung, etc.

→ Das Sicherheitskonzept muss diese Randbedingungen berücksichtigen!

Sicherheitskonzept

- Sicherheitsmaßnahmen kosten Energie durch zusätzliche Rechenzeit und Bandbreite

→ Kompromiss zwischen Grad der Sicherheit und Energieverbrauch nötig!

- Anwendungsspezifischer Kompromiss nötig!

→ möglichst flexibel Sicherheit bereitstellen!

Flexibles Sicherheitskonzept

flexible Sicherheit durch Anforderungsdimensionen

- inhaltsbasierte Sicherheit:
  unterschiedliche Sicherheitsanforderungen verschiedener kommunizierter Inhalte
- situationsbasierte Sicherheit:
  abhängig vom internen Netzzustand und der Umgebung

Nächste Schritte

- Bestimmen der kryptographischen Algorithmen und Konzepte
- genauere Definition von Inhalten und internen Netzzuständen
- Implementierung
- Evaluation

Vielen Dank.

TECHNISCHE UNIVERSITÄT
CAROLO-WILHELMINA
ZU BRAUNSCHWEIG
Institut für Betriebssysteme und Rechnerverbund
http://www.ibr.cs.tu-bs.de

Erste Erfahrungen mit der Karlsruher Sensornetz-Plattform

Erik-Oliver Blaß, Hans-Joachim Hof, Bernhard Hurler, Martina Zitterbart
Institut für Telematik
Universität Karlsruhe
[blass|hof|hurler|zit]@tm.uni-karlsruhe.de

Die fortschreitende Miniaturisierung von Computer-Hardware erlaubt immer kleinere und leistungsfähigere Geräte. In naher Zukunft werden Kleinstcomputer unsere ständigen Begleiter sein, allgegenwärtig unsere Umgebung bestimmen und uns bei alltäglichen Aufgaben unterstützen. Daraus ergibt sich eine Menge vielfältiger Möglichkeiten auf dem Weg dahin aber vor allem auch Hindernisse, die es zu beseitigen gilt. So eignen sich beispielsweise traditionelle Verfahren aus dem Bereich Sicherheit nur sehr eingeschränkt für die angestrebte Zielplattform kleinster eingebetteter Systeme, da dort meist nur wenig Speicher und Rechenleistung zur Verfügung steht. Auch der sehr eingeschränkte Energievorrat der eingesetzten Geräte stellt ein großes Hindernis für die Übernahme traditioneller Architekturen und Algorithmen dar.

Für erste praktische Versuchen mit solchen autonomen Sensoren bzw. Aktoren wurde eine Einheit entwickelt, welche aus einem Mikrocontroller Atmel ATMega 128, einem Bluetooth-Modul und einem oder mehreren Sensoren/Aktoren besteht, die je nach Anwendung direkt auf der Platine oder über eine Drahtverbindung an den Mikrocontroller angeschlossen sind. Zur drahtlosen Kommunikation mit einer Basisstation oder zwischen einzelnen autonomen Sensoren dient das Bluetooth-Modul, für das ein sehr kompakter und effizienter Bluetooth-Stack entwickelt wurde.

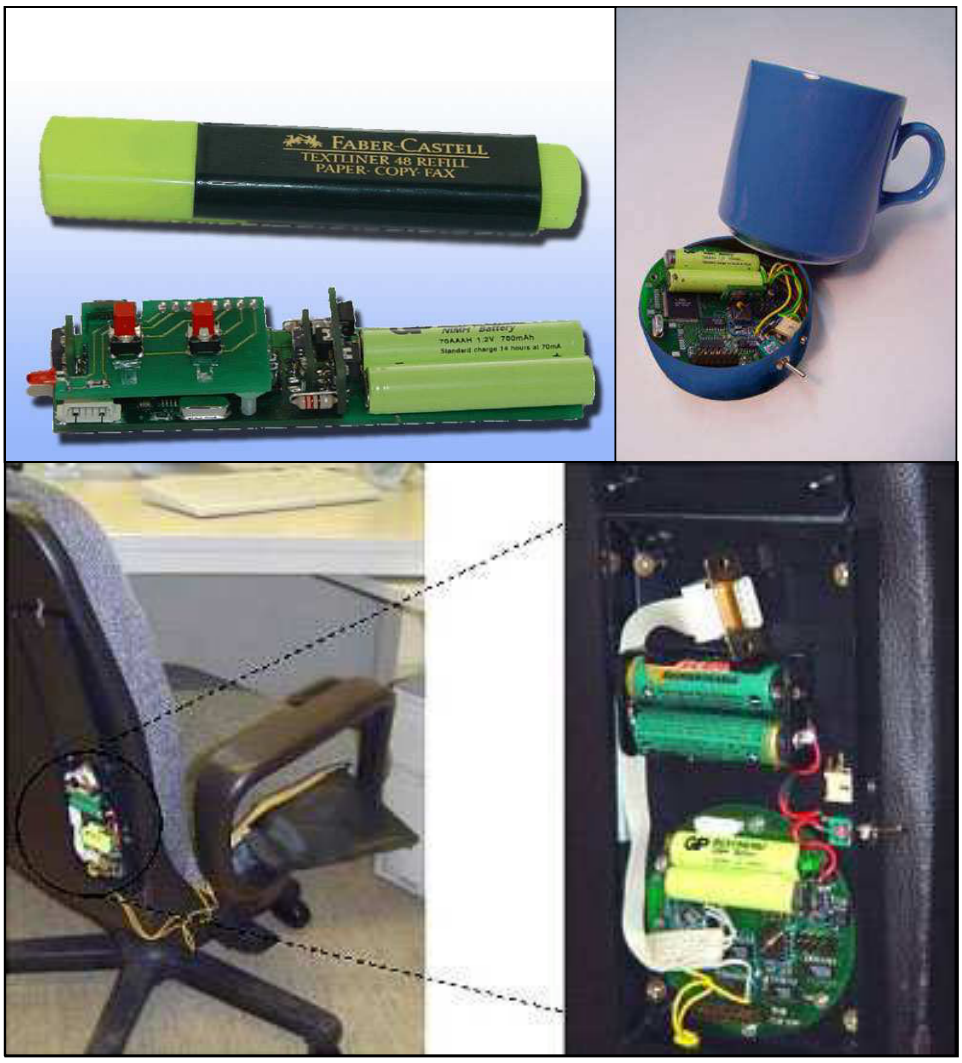


Abbildung 1:Blue Wand (links oben), Blue Cup (rechts oben) und Blue Cushion (unten)

Diese flexible Grundeinheit wird am Institut für Telematik für verschiedenste Anwendungen eingesetzt. Ausgestattet mit Beschleunigungssensoren und Gyroskopen wurde im Rahmen des IEEE-Wettbewerbs 2002 der „BlueWand“ (siehe Abbildung 1) entwickelt, welcher Bewegungen und Gesten des Benutzers aufzeichnet. Der Blue Wand dient somit als alternatives Eingabegerät, mit dem eine Vielzahl von technischen Geräten bedient werden kann. Ein mit einem Drucksensor ausgestatteter Bürostuhl (Blue Cushion, siehe Abbildung 1) wurde in ein CSCW-System (Computer Supported Collaborated Work) integriert. Er dient dort zur Übermittlung von Zustandsinformationen an die Teilnehmer einer Session. Dies ermöglicht non-verbale und intuitive Interaktion zwischen den Teilnehmern. Ein System mit einer größeren Anzahl autonomer Sensoren bzw. Aktoren ist derzeit in der Entstehung. Als Testfall dienen hierzu eine Reihe von Sensoren und Aktoren, welche eine Topfpflanze mit Wasser und geeignetem Licht versorgen sollen, indem Gießanweisungen an den Benutzer und Steuerimpulse an Jalousien übermittelt werden. Als Grundlage für die Elektronik der Topfpflanze dient „Blue Cup“, eine mit Bluetooth vernetzte Kaffeetasse (siehe Abbildung 1), die ihren Füllstand und die Temperatur des enthaltenen Getränks übermitteln kann.

Für die oben aufgeführten Anwendungen existiert bisher ein zentraler Punkt, der mit den verschiedenen Geräten über Bluetooth eine Point-to-Point-Verbindung Kontakt aufnimmt und die von den Sensoren ermittelten Daten zentral auswertet. Im weiteren Verlauf der Arbeit am Institut für Telematik ist geplant, mehr und mehr auf Netzwerke von Sensoren hinzuarbeiten, um sowohl räumlich weiter ausgedehnte Szenarien als auch Systeme ohne zentrale Kontrolle realisieren zu können. Die dazu entworfene Architektur zeigt Abbildung 2. Besonderer Wert wurde beim Systementwurf auf Sicherheitsanforderungen gelegt.

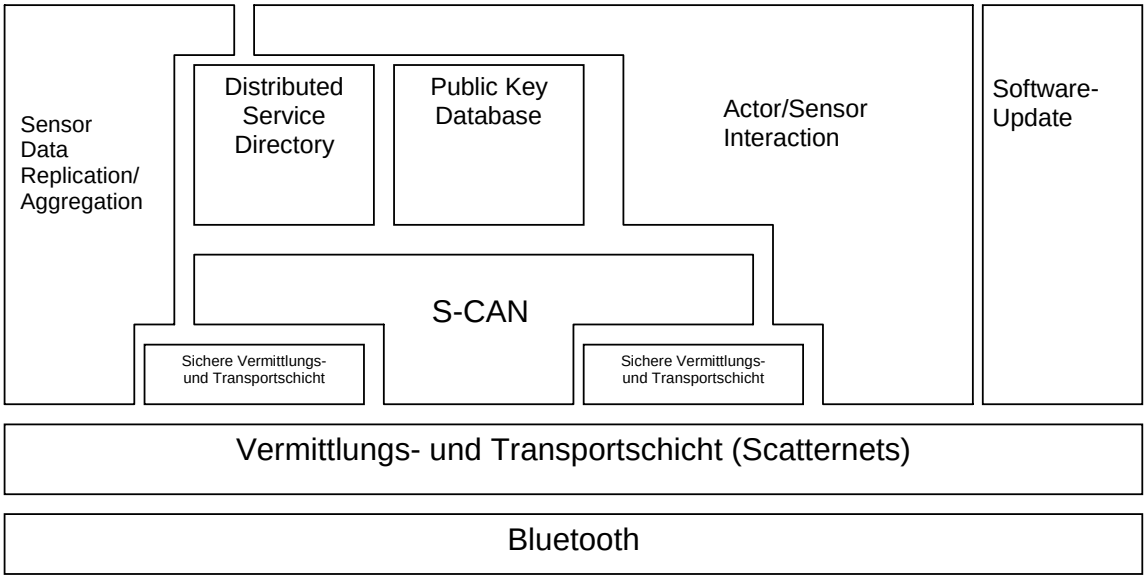


Abbildung 2: Architektur des Sensornetzwerks

Im Folgenden werden die einzelnen Bestandteile der Architektur näher beschrieben:

Sensor Data Replication/Aggregation-Modul:
Im SDR/A-Modul werden Sensordaten vor der Weiterleitung an andere Sensoren aggregiert. Da der mit Abstand größte Teil der Energie eines Sensor für das physikalische Senden von Daten verwendet wird (bzw. in Sensornetzwerken für die Weiterleitung von Paketen), ist es wichtig, möglichst wenig Daten zu übertragen und die anfallenden Daten zu bündeln. Dazu werden redundante Daten nicht weiter übermittelt und Sensordaten von verschiedenen gleichartigen Sensoren miteinander kombiniert, um kürzere Datensätze zu erhalten. Diese Reduktion erfolgt applikationsspezifisch. Das Modul koordiniert ebenfalls die Replikation von häufig angefragten Daten an verschiedene Punkten im Sensornetzwerk. Durch intelligente Replikation kann der Kommunikationsaufwand insbesondere bei periodisch anfallenden Daten erheblich eingeschränkt werden. Grundlage für diese applikationsspezifische Aggregation sind die am Institut für Telematik entwickelten Mechanismen aktiver und programmierbarer Netze.

Software-Update-Modul:
Software-Updates sind für eine lange Lebenszeit eines unzugänglichen, unbeaufsichtigten Netzes wichtig. Die Korrektur von Fehlern, die Erweiterung der Funktionalität ebenso wie die Anpassung an geänderte Umweltverhältnisse werden dadurch ermöglicht bzw. vereinfacht. Software-Updates stellen hohe Anforderungen an die Sicherheit, da die Authentizität und Integrität des zu installierenden Updates eindeutig festgestellt sein muss, bevor es integriert werden kann. Jegliche Sicherheitsoperation muss mit den begrenzten Ressourcen der Zielplattform auskommen. Außerdem darf auch ein misslungenes Software-Update einen Sensor nicht völlig unbrauchbar machen. Dazu ist es nötig, persistente Kernroutinen und Einsprungspunkte für Updates zu definieren und diese auch zu schützen. Updates können als Patch, Multi-Patch (Update Patch von verschiedenen Versionen aus) oder Komplett-Update erfolgen. Dadurch werden Daten bei der Übertragung der Updates eingespart. Jegliche von Sensoren gewonnene Daten werden mit der aktuellen Softwareversion attributiert, so dass durch Anpassungsfunktionen auch dann noch eine Funktionalität des Netzes erbracht werden kann, wenn Daten durch Programmcode verschiedener Versionen gewonnen werden. Angestrebt wird allerdings ein konsistenter Zustand, in dem alle Sensoren die gleiche Softwareversion installiert haben. Inkonsistenzen lassen sich jedoch nicht vermeiden, besonders während eines laufenden Updates. Updates verfügen neben einer Versionsnummer auch noch über einen Hash-Wert aus einer Hash-Kette des Update-Programmierers. Mit diesem Hash-Wert ist es einem Knoten möglich, seinen Nachbarn zu beweisen, dass ein neues Update verfügbar ist. Dadurch kann effizient der Update-Vorgang angestoßen werden und Denial-of-Service-Angriffe werden erschwert. Ziel ist es, dem Update-Programmierer maximale Flexibilität zu gewähren. Die Sensoren am Institut für Telematik der Universität Karlsruhe nutzen die Selbstprogrammierungs-Fähigkeiten des Atmel ATMega128 RISC-Prozessors. Zur Zeit ist bereits die Möglichkeit vorhanden, durch einen Bootloader gezielt Speicherbereiche zu beschreiben. Der Software-Update Code selbst liegt im besonders geschützten Bootloader-Bereich, der nur durch einen speziellen Programmieradapter geändert werden kann. Im laufenden Betrieb ist es so auch im Falle einer misslungene Code-Integration möglich, die Funktionalität des Sensors wiederherzustellen.

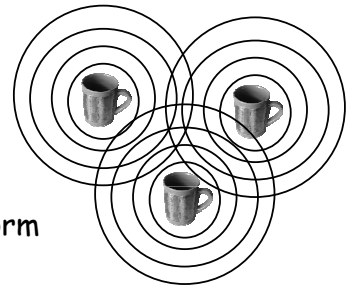
Verteilte Aktor/Sensor-Interaktion:
Im vorliegenden Netzwerkmodell gibt es neben Sensoren, die Daten liefern, auch Aktoren, die aufgrund anwendungsspezifischer Vorbedingungen ihre Umgebung beeinflussen können. Aktoren verbinden sich zeitweilig oder dauerhaft mit Sensoren, wobei Konflikte bei diesem Bindungsvorgang aufgelöst werden müssen. Dazu wurde ein Konzept entwickelt, mit dem im einfachsten Fall Sensordaten abgefragt und Aktoren direkt angesteuert werden können. Für komplexere Zusammenhänge, die ein Zusammenspiel zwischen verschiedenen Sensoren und Aktoren erfordern, besteht die Möglichkeit, dem Sensornetz Aufträge zuzuteilen. Primitive Aufträge beschränken sich auf oben genannte einfache Abfragen oder Steuerungen. Komplexe Aufträge enthalten wiederkehrende bzw. von gewissen Bedingungen abhängige Aufgaben. Diese Aufträge können wiederum aus einer Reihe von primitiven oder komplexen Aufträgen bestehen. Ein Sensornetz erledigt die ihm gestellten Aufträge autonom, ohne dass weitere Eingriffe von außen notwendig werden. Das Auftragskonzept ermöglicht es, ein Sensornetz schnell und ohne Änderung der bestehenden Sensor-/Aktorkomponenten mit neuen Aufgaben zu belegen. Das Auftragskonzept wird momentan in einer vereinfachten Form für den oben genannten „Intelligenten Blumentopf“ implementiert und eingesetzt.

Secure Content Adressable Network:
Einige Sensoren im Sensornetzwerk bilden ein Overlay-Netzwerk, das sogenannte Secure Content Adressable Network (S-CAN). Mit Hilfe dieses Overlays wird ein verteiltes Service Directory mit einer Public Key Database realisiert. Das Service Directory dient primär dazu, verfügbare Daten und Dienste des Netzwerks aufzufinden. Es kann auch zur Datenreplikation durch das SDR/A-Modul verwendet werden. Das Service Directory stellt die Grundlage für das A/SI-Modul dar, das einem Sensor die Möglichkeit bietet, einen Aktor aufzufinden. Die Dienste des Service Directory werden nur gelegentlich genutzt (z.B. zum Aufbau einer Beziehung zwischen einem Aktor und einem Sensor), sind aber von zentraler Bedeutung für die Funktionalität des Sensornetzwerks. Deshalb wurde besonderer Wert auf sichere Konstruktion und Erhalt der Strukur des Overlays gelegt. Dabei verzichtet die vorgeschlagene Lösung auf den Sensoren fast komplett auf asymmetrische kryptographische Verfahren und spart so Rechenleistung und Speicherplatz. Bei der Konstruktion des CAN werden jeweils benachbarte Knoten im Overlay dynamisch mit einem gemeinsamen Geheimnis ausgestattet und können damit fortan sicher und ressourcen-effizient mittels symmetrischer Verschlüsselung kommunizieren. Insbesondere die periodischen Update-Nachrichten über Nachbarschaftsbeziehungen werden durch diese Geheimnis geschützt. Somit wird die Struktur des Overlay-Netzes gesichert. Um neue Geräte in das CAN-Netz aufzunehmen wird ein so genanntes „Master Device“ eingesetzt. Dieses Gerät hat eine herausragende Stellung im Sensornetzwerk: Es bildet den Zugangsschlüssel zum Netzwerk, ist in der Lage auch komplexere kryptographische Operationen auszuführen, ist zustandslos und kommuniziert nur zu bestimmten Zeitpunkten mit dem Sensornetzwerk. Der Einsatz des Master Devices erfordert eine Interaktion mit dem Benutzer und stellt insofern eine Authentifizierung des physikalischen Geräts dar („das ist das Gerät, welches ich gerade aus der Verpackung entnommen ha-

be“). Das Master Device sorgt für eine gleichmäßige Verteilung der Knoten im CAN-Space und stellt den einzelnen Geräten des CAN ein Zertifikat für die von ihnen verwaltete Zone aus. Das Master Device kommuniziert mit dem hinzukommenden Gerät über einen Location Limited Side Channel, im vorliegenden Fall mit physikalischem Kontakt. Außerdem verfügt jedes Gerät über ein gemeinsames Geheimnis mit dem Master Device. Durch ein spezielles Konstruktionsschema muß dies nicht für jedes Gerät auf dem Master Device gespeichert werden. Damit ist sichergestellt, dass neue Geräte auch nach Verlust des Master Devices durch eine Kopie des Master Devices aufgenommen werden können. Der Ausfall eines Geräts und das damit entstehende „Loch“ im CAN-Space kann durch das Master Device mit Hilfe von Bürgschaften der Nachbarn repariert werden. Die im S-CAN zum Einsatz kommenden kryptographischen Algorithmen wurden am Institut für Telematik im Hinblick auf die besonderen Erfordernisse der Zielplattform implementiert. Bisher konnten durch die Algorithmen folgende Leistungsdaten erzielt werden:

- Symmetrische Verschlüsselung über AES, 128 Bit Schlüssellänge: ca. 40 kBit/s
- Asymmetrische Verschlüsselung über Elliptische Kurven Kryptographie, 113Bit Schlüssellänge: ca. 0,06 kBit/s (verschiedene Formen zur erheblichen Optimierung in Arbeit)
- Hash-Algorithmus SHA-1, 4,4 kBit/s (auch hier sind noch einige Optimierungen in Arbeit)

Sämtliche kryptographischen Algorithmen benötigen dabei insgesamt nicht mehr als ca. 900 Bytes RAM.

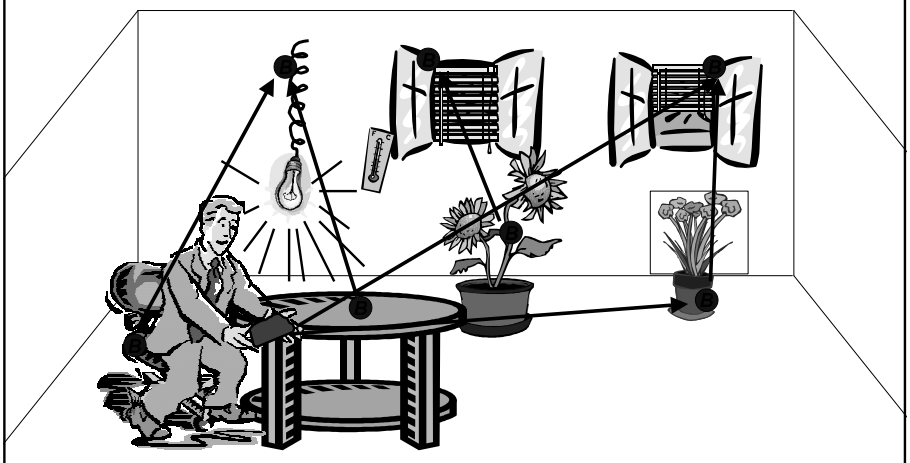


Erste Erfahrungen mit der Karlsruher Sensornetz-Plattform

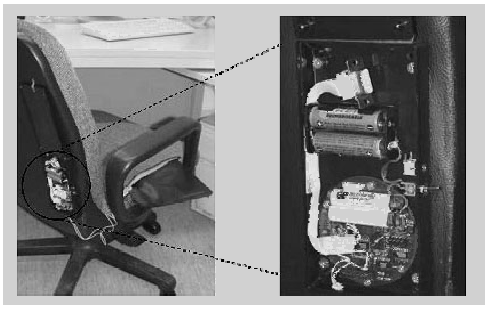
Erik-Oliver Blaß, Hans-Joachim Hof, Bernhard Hurler, Martina Zitterbart

Institut für Telematik
Universität Karlsruhe

Szenario

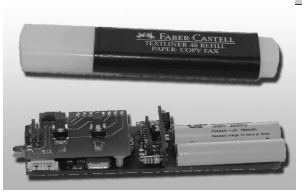


Vorhandene Hardware



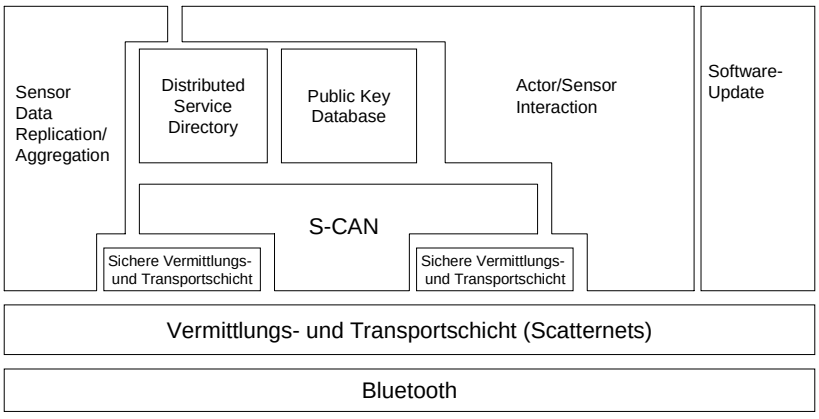
Blue Cushion

Blue Cup



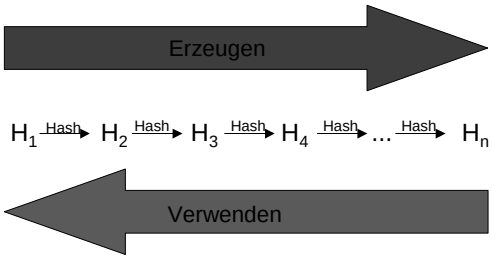
Blue Wand

Architektur



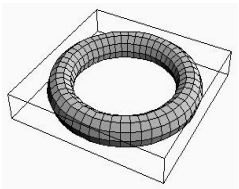
Software Updates

- Für lange Lebenszeit notwendig (Fehlerkorrektur, Umweltanpassung, Funktionserweiterung)
- Daten-zentrische Updates
- Als Patch, Multipatch oder Komplett-Update
- Schutz von Kernroutinen: Update-Code + Connectivity
- Sensordaten attributiert mit Softwareversion
- Anpassungsfunktion für Daten aus verschiedenen Softwareversionen
- Inkonsistenzen nicht zu vermeiden
- Hash-Kette zur Anzeige neuer Software-Updates



Service Directory

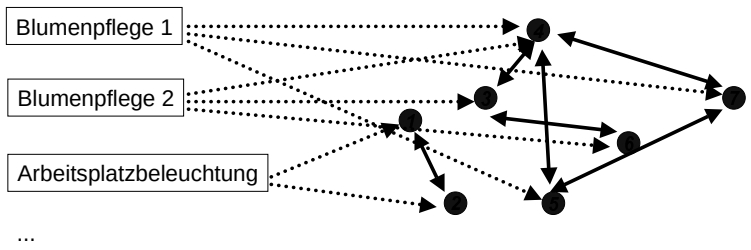
- Content Adressable Network
  - D-dimensionaler Torus
  - Knoten übernehmen einen Teil des CAN-Space
  - Dienstenamen über Hashfunktion abgebildet auf CAN-Space
- Secure Content Adressable Network
  - Erweitert CAN um sichere Konstruktion und Strukturerhalt
  - Größtenteils symmetrische Kryptographie
  - Master Device
- Manche Knoten im CAN
- Diese Knoten können Anfragen von anderen Knoten bearbeiten



Actor/Sensor Interaction

Steuerung eines Sensornetzwerkes durch Aufgaben

- Sensoren/Aktoren – Beteiligung an unterschiedlichen Aufgaben
- Neue Aufgaben - schnelle Implementierung im Netz
- Autonome Bearbeitung der Aufgaben



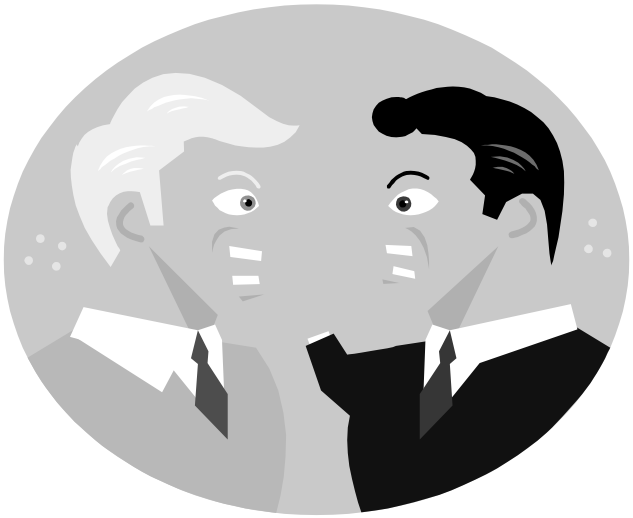
Actor/Sensor Interaction

Aufgabenorientierte Infrastruktur für ein Sensornetzwerk:

- Primitive Tasks
  - QueryTask: Daten von Sensoren
  - OrderTask: Steuerung von Aktoren
- Complex Tasks
  - ConditionalTask: bedingte Ausführung von Aufgaben
  - RepetetiveTask: wiederholte Ausführung von Aufgaben

-> Rasche Aufgabenstellung und -änderung möglich
-> Sensor-/Aktor-Einheiten können sich an vielen Aufgaben beteiligen
-> Autonome Aufgabenerledigung

Diskussion



BTnodes - Applications and Architecture Compared

Jan Beutel, Oliver Kasten, Matthias Ringwald *
Swiss Federal Institute of Technology (ETH) Zurich
8092 Zurich, Switzerland
beutel@tik.ee.ethz.ch

Abstract

We motivate a prototyping platform for ad-hoc networking research showing some requirements and constraints. The architecture of the BTnodes, each of which can store information, compute and communicate, is explained in conjunction with some demo applications that have been implemented. Important requirements and design trade-offs to be able to support multiple, compatible communication interfaces, to handle limited resources, for power-aware operation and for efficient testbed deployment are discussed.

1 Introduction

By using standardized communication interfaces, wireless sensor networking nodes can interact with these everyday appliances, peripheral devices, sensors and actors alike. According to [3, 6] and others, services in the network are the dominating factor for future growth. Fostering this interaction are well established and acquainted user interfaces on already common devices such as PDAs and cellular phones that make it possible to reach out into the digital representation of smart everyday objects and interactions.

Typical applications in research are in fast prototyping of demo applications [2, 7], interfacing to other devices (sensors, actors, multimedia and computing devices) [12] and the realization of emerging networking concepts that have so far only been theoretically specified and simulated.

Bluetooth is a connection-oriented, wireless communication medium that assures interoperability between different devices and enables application development through a standardized interface. This Host Controller Interface (HCI) hides most of the lower-layer abstraction from the system developer and leaves host-system resources to higher-layer applications. Networks of Bluetooth devices are organized in Piconets. These are Master-Slave star topologies that can be interconnected to form larger Scatternets. Compared to other media used in low-power wireless research [11, 9, 4], Bluetooth offers a host of high-level link-layer functionality such as multiplexing, integrated audio, different channel characteristics, link keys and encryption. The most apparent difference is that the developer is not dealing with a baseband and MAC interface but with dedicated communication channels. Applications thus need to be wireless-aware (broadcast medium), but no knowledge of digital signal processing and real-time systems is necessary.

Key requirements for a ubiquitous research platform are flexibility, power-aware operation, efficient deployment and the support of standardized interfaces. The following target features have been realized in the design of the BTnode:

- In-circuit programmable Bluetooth platform
- Remote update of system software
- Low component count
- Compact overall system size
- Simple debugging capability
- Sensor and user interface
- Single voltage design with power management

2 BTnode Architecture

The BTnode is an autonomous wireless communication and computing platform based on a Bluetooth radio module and a microcontroller. The benefit of this

*The work presented in this paper was supported by the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center supported by the Swiss National Science Foundation under grant number 5005-67322.

platform is having a small form factor of 6x4 cm while still maintaining a standard wireless interface. With its general purpose interfaces the BTnode can be used with many peripherals, such as sensors, actors, DSPs, serial devices (like GPS receivers, RFID readers, etc.) and user interface components.

The BTnode hardware (see Fig. 1) consists of an Atmel ATmega128L microcontroller with on-chip memory and peripherals. The microcontroller features an 8-bit RISC core delivering up to 8 MIPS at a maximum of 8 MHz. The on-chip memory consists of 128 kbytes of in-system programmable Flash memory, 4 kbytes of SRAM and 4 kbytes of EEPROM. There are numerous peripherals integrated as well: JTAG for debugging, timers, counters, pulse-width modulation, 10-bit analog-digital converter, I2C bus, two hardware UARTs. An external low-power SRAM adds an additional 240 kbytes of data memory to the BTnode system.

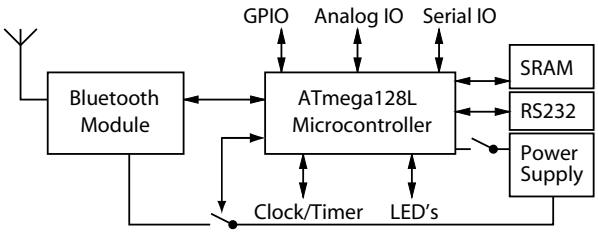


Figure 1. The BTnode system overview

A real-time clock is driven by an external quartz oscillator to support timing updates while the device is in low-power sleep mode. The system clock is generated from an external 7.3728 MHz crystal oscillator.

An Ericsson Bluetooth module is connected to one of the serial ports of the microcontroller using a detachable module carrier and to a planar inverted F antenna (PIFA) that is integrated into the board to further reduce the critical components. The operating and power modes of the Bluetooth module can be controlled by the MCU.

Four LEDs were directly integrated, mostly for the convenience of debugging and monitoring, although they could easily be optional external add-ons. One analog line is connected to the battery input and allows to monitor the battery status. The other peripherals are directly accessible via external connectors. These connectors all have the same setup with 4 signal and additional power and ground lines to be able to flexibly power and control add-on sensor peripherals. The supply voltage for MCU, memory and Bluetooth is fed through separate 0 Ohm resistors. When replaced by a current meter, this is a common way to enable exact in-situ power-consumption profiling for each component.

3 Communication Oriented OS Support

The BTnode system software is a lightweight operating system made up of low-level drivers that are interrupt driven and a simple dispatcher for scheduling multiple threads. This OS is well-suited for the applications of such small-scale networking devices that will consist mostly of simple IO and monitoring tasks and communication.

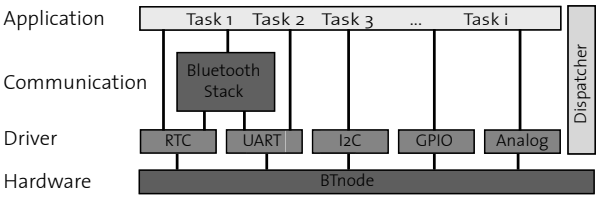


Figure 2. A lightweight communication oriented OS framework for WSN applications

An event-driven programming model provides convenient functions for resource management. The dispatcher is used for the scheduling of tasks; it implements coarse-grained cooperative multithreading. Only one task (event handler) can be active at a time. Events are processed in the order they appear. Every event handler is always completely executed until the next one is scheduled. So every event handler depends on the previous event handler to terminate in time. A software component, such as a driver, can generate an event to notify other components of the occurrence of a change in state that requires further action.

The second part of the system software are low-level interrupt-driven device drivers that allow applications to access the peripherals and interfaces in a standard way. The drivers are designed with fixed buffer lengths that can be adjusted at compile time to meet the stringent memory requirements. Available drivers include memory, real-time clock, UART, I2C, LEDs, power modes and AD converter.

4 Power Aware Operation

Different power-saving modes are available for both the microcontroller and the Bluetooth module. A real-time clock controlled by a separate driver is driven by a separate oscillator to allow to make use of the

low-power modes of the BTnode over longer periods of time. Furthermore, the microcontroller can be run at different operating frequencies controlled by the software.

A simple qualitative sensor application example (see Table 1) with a 10 % duty cycle reveals a quite acceptable average power-consumption of 6.5 mW and a battery lifetime T on the order of weeks on a standard 840 mAh Li-ion battery. Newer Bluetooth hardware is much less power-hungry than our first generation developer hardware, reducing power-consumption in communication mode by a factor of 2-4.

Operation	**t [sec]**	**P [mW]**	**T [h]**
Sensing	4	12	252
Communication	2	160	19
Idle	54	0.5	6048
Total		**6.5**	**421**

Table 1. Power consumption example

5 Platform Deployment and Tools

A software kit consisting of a build environment (avr-gcc cross compiler and standard libraries), source code, debugging support, demo examples and documentation has been assembled for the BTnodes and is available for download. A support mailing list and software repository are also available to developers.

In order to support fast prototyping and debugging without having to download to the embedded target on every design iteration, a separate build tree on Linux with the required interface and hardware emulation has been developed. This allows immediate execution of the identical system software as on the embedded target on a conventional PC with an attached serial Bluetooth device. A further advantage of this emulation mode is the possibility to read and write files on the host system and use the host system's advanced computing resources, e.g. for data gathering and analysis in a network of sensor nodes.

To simplify maintenance and application development with many BTnodes, a mechanism allowing selective network flooding with program-code updates has been developed.

The BTnodes have been developed and distributed in cooperation of the NCCR-MICS [5, 1] and the Smart-Its Project, the latter being a part of the EU Disappearing Computer initiative. The low complexity and small bill of material of the BTnodes results in a unit cost of USD 110 for the initial deployment of currently 200 units that have been distributed among different research groups worldwide.

6 Applications

Many applications have been realized using the BTnodes. Most of them are ubiquitous-sensing and monitoring applications where fast prototyping and ease of deployment are a key concerns [2, 10]. Other applications such as the one shown in Fig. 3 want to interface networks of sensors to commercial devices such as cell phones [12, 13] and PDAs [7]. For the latter type of application, compatibility and adaptability of the interfaces is the most critical issue.

Figure 3. Product monitoring using BTnodes as smart tags and SMS via mobile phones

Other applications are more specific in the networking requirements and deal with multihop schemes, adhoc routing, positioning and topology discovery, and hardware-adaptable systems [8].

References

[1] NCCR-MICS: Mobile Information and Communication Systems, Terminodes. http://www.terminodes.org.

[2] S. Antifakos, F. Michahelles, and B. Schiele. Proactive Instructions for Furniture Assembly. In *Proceedings of the The Fourth International Conference on Ubiquitous Computing (UbiComp 2002)*, Goeteborg, Sweden, September 2002.

[3] B. Raman et al. The SAHARA Model for Service Composition Across Multiple Providers. *Pervasive Computing*, August 2002.

[4] J. Hill, R. Szewczyk, A. Woo, S Hollar, D.E. Culler, and K.S.J. Pister. System architecture directions for networked sensors. In *Architectural Support for Programming Languages and Operating Systems*, pages 93–104, 2000.

[5] P. Hubaux-J, T. Gross, Y. Le-Boudec-J, and M. Vetterli. Toward self-organized mobile ad hoc networks: The Terminodes Project. *IEEE Communications Magazine*, 39(1):118–124, Jan. 2001.

[6] D.G. Leeper. A Long-Term View of Short-Range Wireless. *IEEE Computer*, 34(6):39–44, June 2001.

[7] F. Michahelles and B. Schiele. Better rescue through sensors. In *Proceedings of First International Workshop on Ubiquitous Computing for Cognitive Aids*, Goeteborg, Sweden, September 2002.

[8] C. Plessl, R. Enzler, H. Walder, J. Beutel, M. Platzner, and L. Thiele. Reconfigurable hardware in wearable computing nodes. In *Proc. Int. Symp. on Wearable Computers (ISWC'02)*, pages 215–222. IEEE, October 2002.

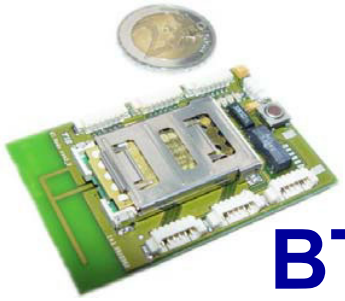
[9] J. Rabaey, J. Ammer, J. da Silva Jr., D. Patel, and S. Roundy. PicoRadio Supports Ad Hoc Ultra-Low Power Wireless Networking. *IEEE Computer*, 33(7):42–48, July 2000.

[10] K. Römer. The lighthouse location system for smart dust. In *Proceedings of the First ACM/USENIX Conference on Mobile Systems, Applications, and Services (MobiSys 2003)*, May 2003.

[11] E. Shih, P. Bahl, and M. Sinclair. Wake on wireless: An event driven energy saving strategy for battery operated devices. In *Proceedings of ACM MobiCom 2002, Atlanta*, September 2002.

[12] F. Siegemund. Spontaneous interaction in ubiquitous computing settings using mobile phones and short text messages. In *Workshop on Supporting Spontaneous Interaction in Ubiquitous Computing Settings, Ubicomp 2002*, September 2002.

[13] F. Siegemund and C. Flörkemeier. Interaction in Pervasive Computing Settings using Bluetooth-enabled Active Tags and Passive RFID Technology together with Mobile Phones. In *Proceedings of the IEEE International Conference on Pervasive Computing and Communications (PerCom 2003)*, March 2003.



BTnodes

Applications and Architecture Compared

Jan Beutel, Oliver Kasten, Matthias Ringwald

RESEARCH GROUP FOR
Distributed Systems

TIK
Computer Engineering and Networks Laboratory

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Initial Projects

Smart everyday objects

by attaching sensor nodes:

- self aware
- context sensitive
- cooperative
- integration into computing environment

Ad hoc networking scenarios

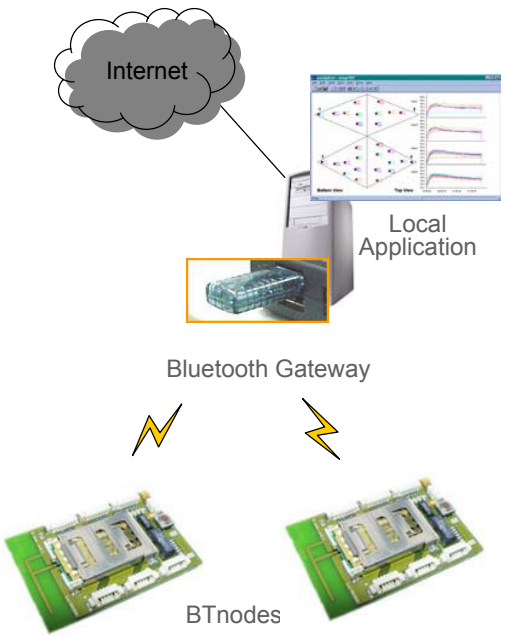
- integrated application protocols
- scalable multi-hop routing

Wearable Computing

smartoits

Consumer Electronics Integration

Backend Connectivity



Connectivity to

- application servers
- other networks

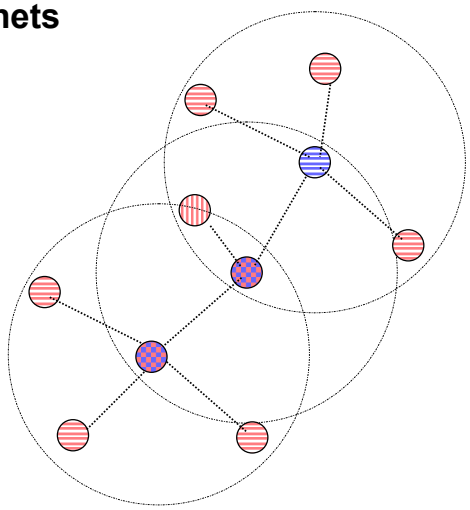
Clusters of mobile networks

- using GSM
- using SMS services
- Wireless LAN
- interfacing to other sensor networks

Bluetooth Piconets

Communication organized in piconets

- controlled by one master
- up to 7 active slaves
- 255 inactive (parked) slaves



Master-Slave

- implements centralized control
- synchronization of all slaves
- only master-slave communication

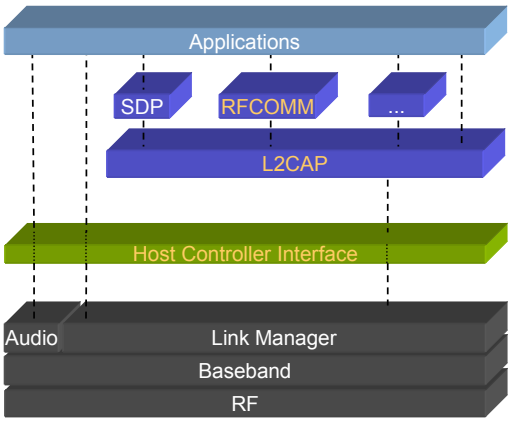
Multiple piconets

- separate channels
- no coordination

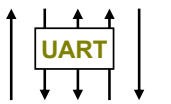
Bluetooth Host Controller Interface

- standard interface for protocol software
- providing access to lower levels of the protocol stack

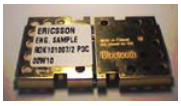
HCI_COMMAND
HCI_EVENT



Host processor



Physical interface

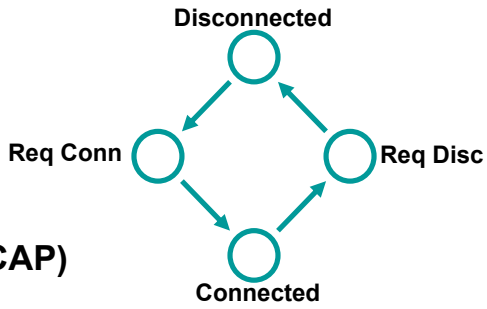


Bluetooth module

Bluetooth Connections

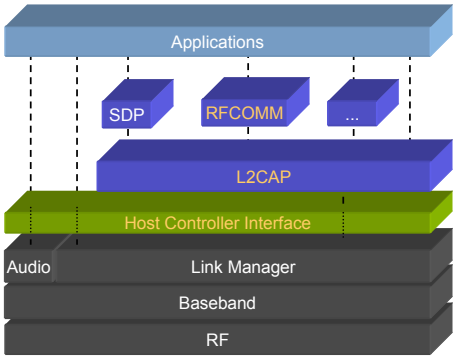
Managed by the host controller

Statemachine for each connection



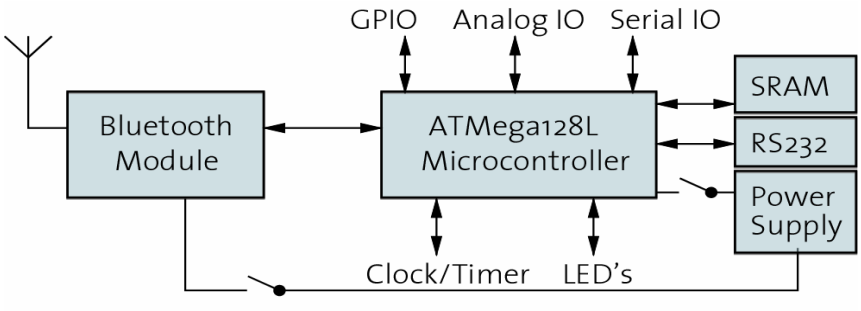
Link Layer Control & Adaptation (L2CAP)

- connection-oriented
- connectionless data
- protocol multiplexing for a single "air interface"
- packet segmentation and reassembly
- channel abstraction
- encryption
- security
- …



Hardware Requirements

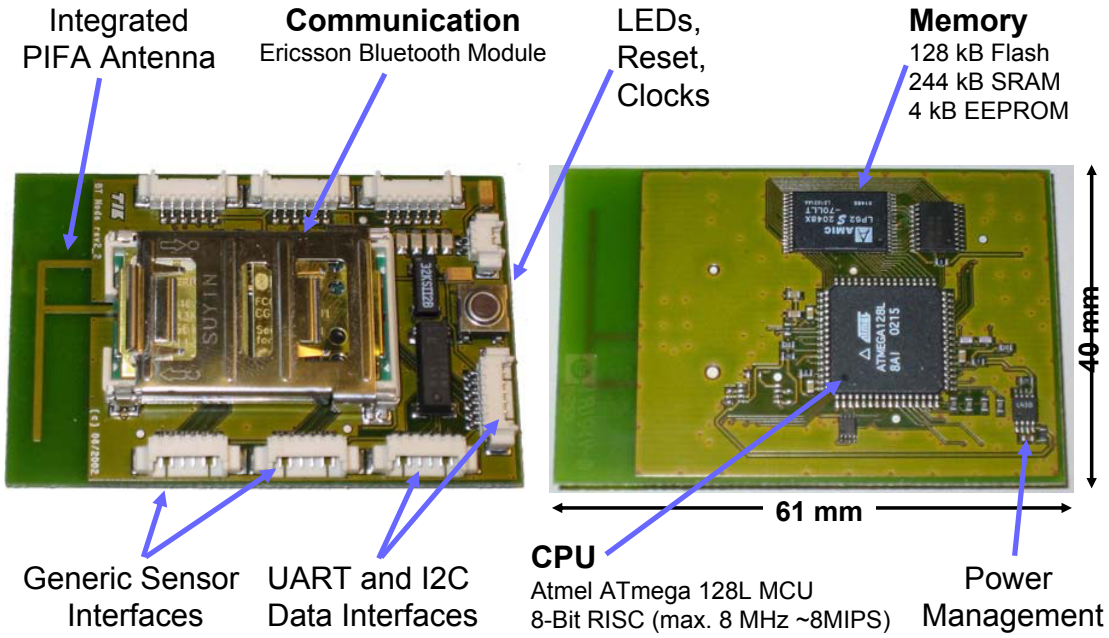
Autonomous wireless communication and computing platform based on a Bluetooth radio module and a microcontroller.



Requirements

- small form factor, low component count
- standardized wireless interface
- flexible and cost effective deployment of large quantities of networking nodes

Hardware Details



Designing for Power Aware Operation

Features

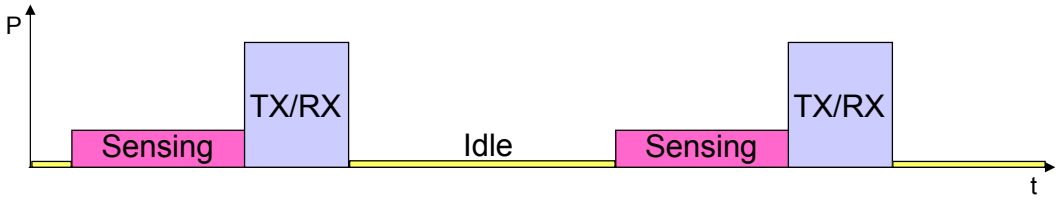
- optional switchable power supply for Bluetooth module
- MCU with 6 power down modes, low idle/sleep current
- frequency scaling: 7.3 MHz - 57 kHz

- single power supply (3.6 – 16 V), single internal voltage (3.3 V)
- battery charge indicator
- direct current access shunts for all components
- internal Vcc available at every connector to power external sensor modules

Power consumption @ 7.3 MHz [mW]	**max**	**typ**	**Lifetime [h]***
– Bluetooth Connected/CPU On	250	160	12-19
– Bluetooth Idle/CPU On	95	67	32-45
– Bluetooth Off/CPU Idle	15	12	202-252
– Bluetooth Off/CPU Sleep	6	<0.5	504-6048

*on 840 mAh Li-ion

Power Consumption Details



Sensor Network Example: 10% duty cycle

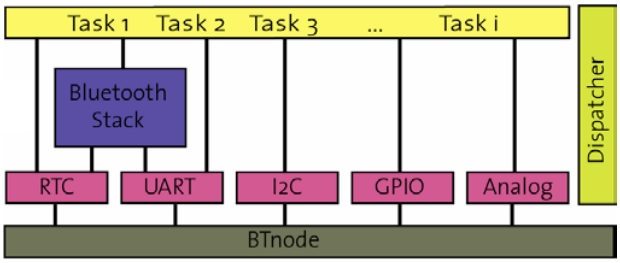
Operation	Power consumption [mW]	Lifetime [h]*
4 sec sensing	12	252
2 sec communication	160	19
54 sec idle	0.5	6048
Total duty cycle	~ 6.5 mW	421

*on 840 mAh Li-ion

System Software

Lightweight OS

- event-driven application model
- cooperative multithreading
- device drivers (UART, RTC, ADC, ...)

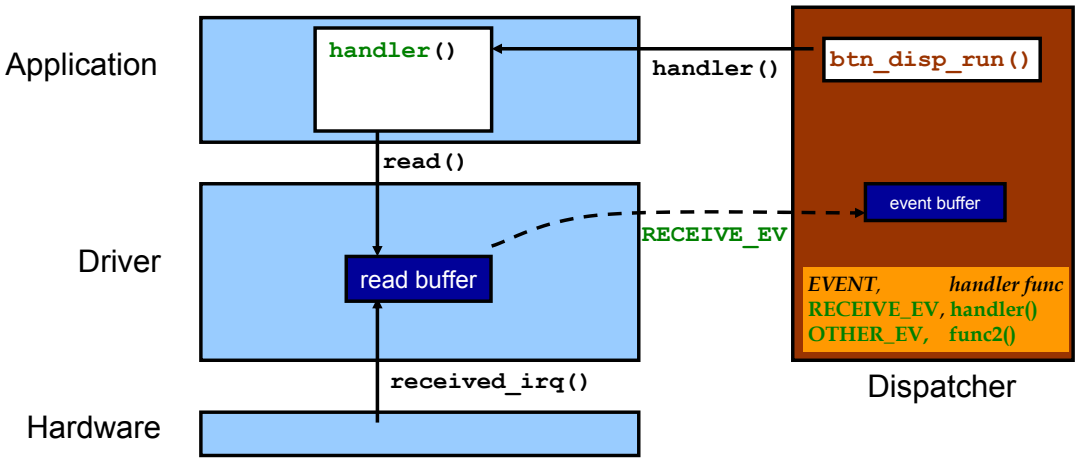


Programming

- standard C language
- high-level Bluetooth interface
- system software available as library
- emulation environment on Linux

Multiple Drivers and the Dispatcher

```
void handler( /* ... */ ) {}
void main() {
   btn_disp_ev_reg( RECEIVE_EV, handler, 0 );
   btn_disp_run();
}
```



Better Avalanche Rescue through Sensors

heart rate, oxygen saturation

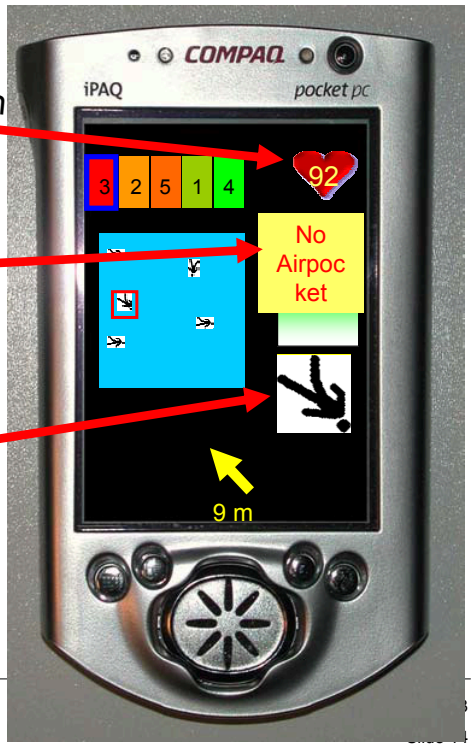
oximeter

air pocket detection

oxygen sensor

orientation, mobility

accelerometer, inclination



Bluetooth enabled Appliances

Communication with other Bluetooth enables devices

– standard Bluetooth profiles for SMS, object push and RFCOMM

BTnode enabled Egg Carton

SMS from Egg Carton

Interactive Dialog

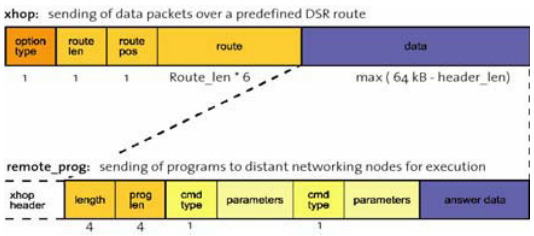
XHOP/R-DSR Multihop Network

Bluetooth multihop source routing prototype

– integrated scalable application protocol

– based on Dynamic Source Routing (CMU)

– routing across piconet borders to support >8 nodes

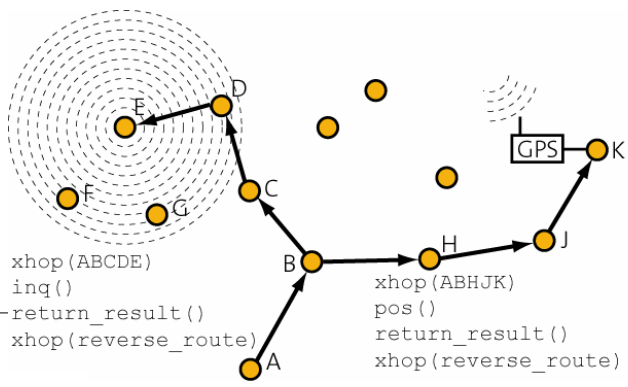


Remote topology discovery

– script like command language in the payload

Performance

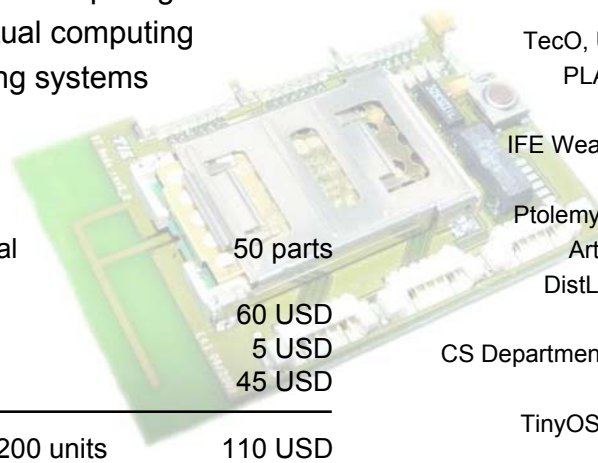
– 1-2 sec per hop, depending on inquiries



Other Projects using BTnodes

200 units with 16 research groups

- smart objects
- routing
- wearable computing
- perceptual computing
- operating systems



Bill of material	50 parts
Parts	60 USD
Assembly	5 USD
Bluetooth	45 USD
Unit cost @ 200 units	110 USD

VTT, FI
DSG, ETH Zurich, CH
PCCV, ETH Zurich, CH
TecO, University of Karlsruhe, GE
PLAY, Interactive Institute, SE
TIK, ETH Zurich, CH
IFE Wearable Lab, ETH Zurich, CH
NTT DoCoMo, Munich, GE
Ptolemy Group, UC Berkeley, USA
Art of Technology, Zurich, CH
DistLab, Diku, Copenhagen, DK
LAP, EPF Lausanne, CH
CS Department, Lancaster University, UK
LSL, EPF Lausanne, CH
TinyOS Group, UC Berkeley, USA
University of Uppsala, SE

Hands-on experience

From tool installation to first application in less than a day

20 student projects completed

Bootcamp held twice for related research projects

Sensor Hardware for Real-World Experiments

Hartmut Ritter, Thiemo Voigt, Achim Liers, Jochen Schiller
Freie Universität Berlin, Germany
Institut für Informatik
{voigt, hritter, liers, schiller}@inf.fu-berlin.de

1 Introduction

Research in the area of sensor networks is often conducted using simulations. Though this is necessary for studies of scalability issues, we argue in this paper that experiences with real hardware help to develop new concepts. Real-world experiments help to reveal limitations and to integrate them in future large-scale simulations. We present a sensor hardware that is part of our Scatterweb project and discuss the application of transmission power adjustment for topology discovery. We plan to make the nodes available for other research groups soon.

2 Hardware Description

One of the design goals of the sensor nodes we built in our lab was to provide a small, yet universally usable platform. Therefore, we integrated a lot of different sensors that could be omitted in special scenarios. The sensors provided at the moment are:

- A light sensor for the detection of visible light
- A passive infrared sensor for movement detection
- A temperature sensor
- A gravitation sensor for the detection of movement of the sensor board
- A microphone for determination of the ambient noise level

All these sensors are connected with a controller, the Texas Instruments MSP 430. It handles all sensor data and coordinates communication with peer nodes. In addition, an external timer module is provided. With this sensor node three different power modes can be deployed. In a deep-sleep state the controller is basically shut down and wakes up only when a timer interrupt or an infrared signal occurs. In normal standby mode (processor up and running) the currency increases from 60 µA in the deep-sleep state to about 1,6 mA. Even in the most power-intensive mode (currencies of 40 mA) the node can be run with a solar cell.

Communication takes place via an RF module in the 868 MHz band. The RF module (TR1001 from RFM) provides an interesting feature as it allows currency-controlled regulation of the transmission power. Using a digital resistor the controller can regulate transmission power in 100 steps (we call these power control values) between off and maximum power.

Figure 1: Sensor Node

3 Initial experiments with transmitter output voltage

As mentioned in the previous section, the 868 MHz transceiver allows us to adapt the transmission range by software. This is useful in at least two scenarios:

- to inquire information about the location of sensors and the topology of the network
- to simplify the realization of experiments with multi-hop communication between sensor nodes

While simulations often assume well-defined transmission ranges, the transmission range of a sensor depends very much on the actual environment. For example, as some initial experiments have shown, the maximum transmission range outside is about 300 meters while it is about 100 meters inside a building through one or two walls. Further experimental data is shown in Figures 2 to 4. Figure 2 shows the transmission range dependent on the power control values in the case of an outdoor measurement. Note that the maximum transmission range is much lower than the 300 meters mentioned above since the measurements were performed closer to the ground. Higher transmission ranges were achieved by holding up the sensors in the air. The data shown in Figure 3 was collected with sensors placed in the corridor while the results in Figure 4 are from measurements with sensors on a board inside the hardware lab.

From these results we are going further into two directions:

- It will be possible to set up sensor networks of approx. 20-30 nodes on a table, as the adjusting of the transmission power allows building extremely short-ranged connections between nodes. Real-world experiments with large-scale sensor networks thus do not require large rooms.
- Second, topology discovery using triangulation will be possible. We are currently investigating into this. We assume that precision will not be as good as with transmission delay measurements, though on the other hand the measurement results of a large number of nodes in a sensor network can be correlated.

4 Other Work

Within the scope of project seminars and thesis works students are currently realizing several real world experiments in different areas:

- Data-centric routing
- Access to sensor data over the Internet
- Alarm notification and management of nodes using GSM’s short message service
- Integration into database

Appendix: Transmission Range Measurements

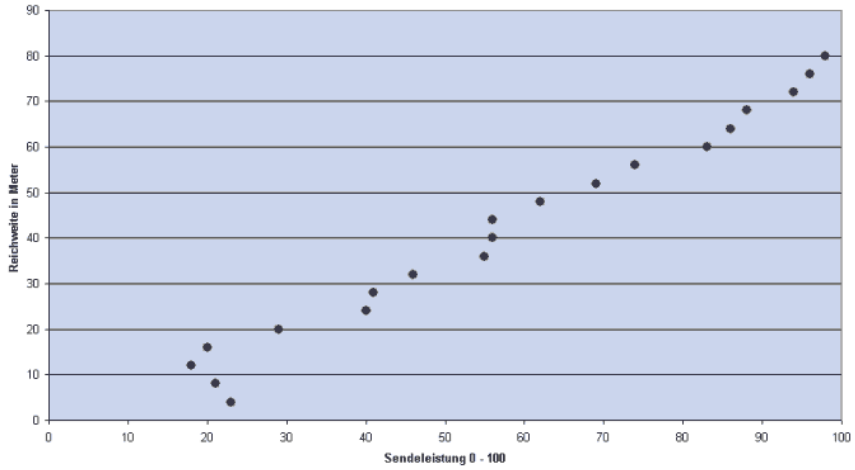


Figure 2: Transmission Range Outside

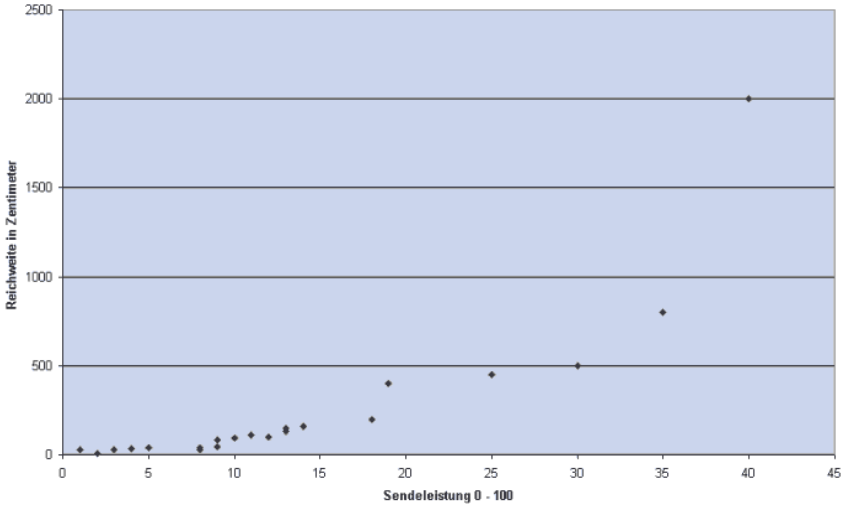


Figure 3: Transmission Range Inside, in a Corridor

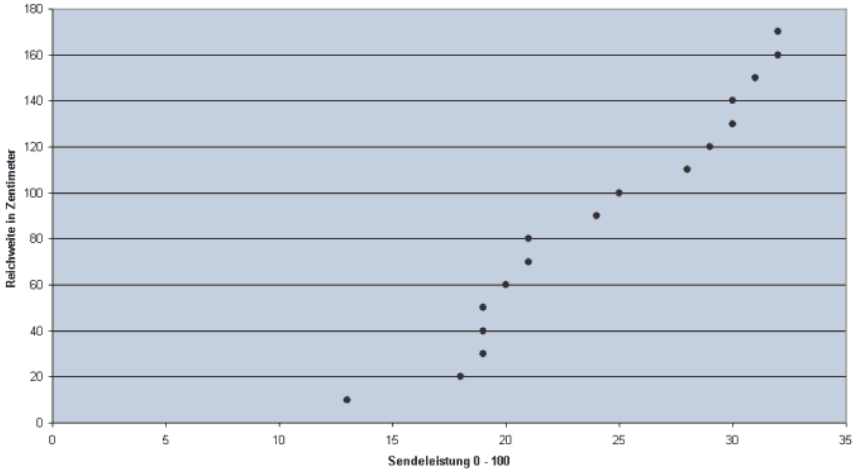


Figure 4: Transmission Range Inside on a Table

Betrachtungen zum Energieverbrauch von Sensornetzknoten

Katja Schwieger, Heinrich Nuszkowski and Gerhard Fettweis
Vodafone Stiftungslehrstuhl Mobile Nachrichtensysteme
Technische Universität Dresden, Mommsenstr. 18, D-01062 Dresden
Email: {schwieg, nuszkows, fettweis}@ifn.et.tu-dresden.de, Tel.: +49 351 463 33919, Fax: +49 351 463 37255

I. Einführung

Drahtlose Sensornetzwerke stellen eine neue Herausforderung auf dem Gebiet der Mobilkommunikation dar. Im Gegensatz zu herkömmlichen Funknetzen spielt dabei die verfügbare Bandbreite kaum eine Rolle. Da die Netzknoten im Allgemeinen batteriebetrieben sind, muss vielmehr der Energieverbrauch der Sensoren optimiert werden. Um energieeffiziente Protokolle zu entwickeln, muss zunächst der Energieverbrauch der Hardwarekomponenten in verschiedenen Zuständen ermittelt werden. Auch wenn die Netzwerke zumeist multi-hop Funktionalität aufweisen, ist es sinnvoll, zunächst den Energieverbrauch im single-hop Fall zu untersuchen. Ausgehend von einer an unserem Lehrstuhl entwickelten Hardware soll im Folgenden der Energieverbrauch in einem single-hop Netzwerk untersucht werden. Dabei werden die Modi "Senden" (TX), "Empfangen" (RX) und "Sleep" separat betrachtet. Mit Hilfe der Analyse können wichtige Rückschlüsse auf das Design des Protokolls gezogen werden. In der Fachliteratur werden bisher zumeist Protokoll-Probleme für energieeffiziente Übertragung behandelt (z.B. [1], [2]), nur wenige Veröffentlichungen behandeln die Bitübertragungsschicht (z.B. [3], [4]). Dies ist ein Beitrag, der die schichtenübergreifende Betrachtung (OSI-Modell) bei der Energieoptimierung anstrebt.

II. Aufbau des Sensornetzwerkes

Abbildung 1 zeigt den prinzipiellen Aufbau eines Sensorknotens. Er besteht aus dem eigentlichen Sensor, einem Digitalteil (DSP oder μC), das das Protokoll enthält und die Basisbandverarbeitung durchführt und dem Analogteil, welches das digitale Signal auf die Trägerfrequenz f_c moduliert. Nicht dargestellt sind u.a. die Batterie, der Leistungsverstärker (power amplifier-PA) und diverse Schnittstellen.

In unserem Fall ist das Digitalteil ein μC von Texas Instruments (TI) und das Analogteil der CC1000 von Chipcon, der den PA bereits integriert. Tabelle I zeigt den Stromverbrauch der Komponenten im TX/RX/Sleep-Mode. Der Verbrauch des Sensors wird nicht in die Berechnungen einbezogen, da er stark vom jeweiligen Senor abhängt. Der Gesamtverbrauch wurde direkt am Sensorknoten gemessen und enthält also auch den Verbrauch externer Bauelemente.

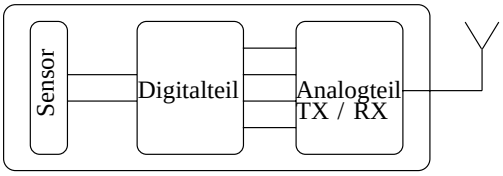


Abb. 1. Aufbau eines Sensorknoten

	TX [mA] P=-20dBm...10dBm	RX [mA]	Sleep [μA]
TI μC	0.5	0.5	1.6
CC1000	5.3...26.7	7.4	1
gesamt	>5.8-27.2	14	6

Tabelle I
Stromverbrauch der Knotenkomponenten

Folgendes single-hop Szenario wird nun untersucht: Ein Sensor sendet Daten über einen AWGN-Kanal an eine 30m entfernte Basisstation (BS). Dabei wird das Paket durch den Kanal beeinträchtigt; außerdem kann es zu Kollisionen mit anderen Paketen kommen. An der BS wird zunächst die Synchronisationssequenz gesucht (=Detektion). Nach erfolgreicher Detektion wird das Signal demoduliert und in einem letzten Schritt der Fehlerschutz dekodiert. Werden alle 3 Stufen erfolgreich abgeschlossen, sendet die BS ein Acknowledgment (Ack). Dessen Ankunft am Sensor wird als erfolgreich angenommen, da die BS über ausreichende Energieressourcen verfügt, um das Ack mit genügend hoher Sendeleistung zu übertragen. Bekommt der Sensor kein Ack, wird das Paket so lange übertragen, bis die Anzahl der maximalen Wiederholungen L erreicht ist.
Weitere Randbedingungen sind: Das Netzwerk besteht aus 100 Sensoren, wobei jeder Sensor alle 5 Minuten ein Datum sendet. Die unkodierte Paketlänge beträgt 128 bit. Der Kanalzugriff erfolgt mit slotted ALOHA [5]. Als Modulationsschema wird 2 FSK (nichtkohärente Detektion) eingesetzt. Es wird im 433 MHz-ISM-Band mit einer Datenrate von 2.4 kbps gearbeitet. Diese Spezifikation sind mit denen im IEEE 802.15.4.-Standard [6] vergleichbar.
Der Übertragungsprozess wird nun mit Hilfe von Markoff-Ketten und den dazugehörigen Signalflußgraphen analysiert. Damit lassen sich u.a. berechnen: Die durchschnittliche Energie, die bei gegebenenem L für eine Übertragung benötigt wird, die mittlere Anzahl benötigter Wiederholungen sowie die Wahrscheinlichkeit für die erfolgreiche Ankunft eines Paketes an der BS.

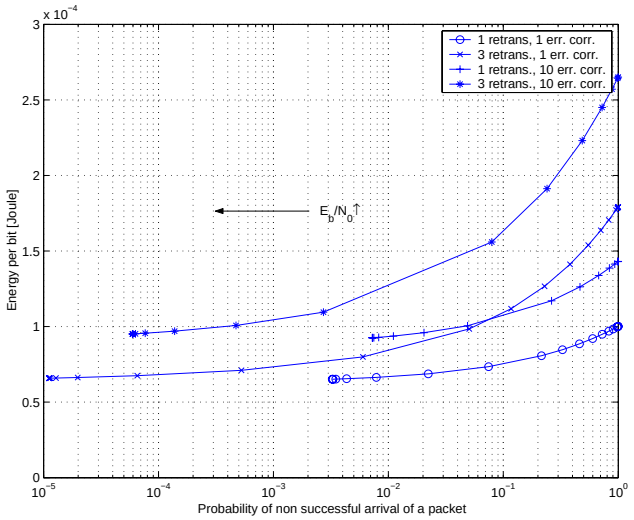


Abb. 2. Gesamtenergieverbrauch vs. Nichtankunftswahrscheinlichkeit

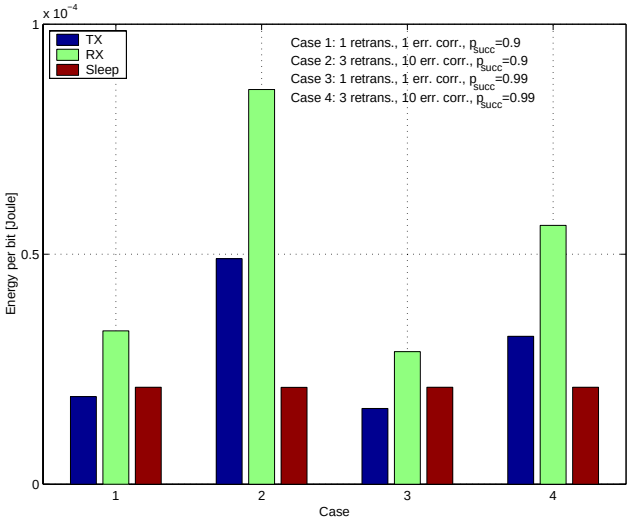


Abb. 3. Energieverbrauch der Einzelkomponenten

III. Ergebnisse

Zunächst soll der Gesamtenergieverbrauch als Funktion der (nicht-)erfolgreichen Übertragung eines Datenpaketes betrachtet werden.

Folgende Szenarien werden dabei in einer ersten Untersuchung betrachtet: Die Anzahl der wiederholten Übertragung eines Paketes bei nicht erfolgreicher Ankunft ist auf 1 bzw. 3 beschränkt. Es werden BCH-fehlerschutzkodierte Pakete versendet, die entweder 152 bit lang sind (1 Fehler korrigierbar) oder 252 bit enthalten (10 Fehler korrigierbar). Abbildung 2 zeigt den Energieverbrauch pro Datenbit bei gegebener Wahrscheinlichkeit für eine erfolglose Übertragung. Folgende Erkenntnisse lassen sich daraus ableiten:

- Eine höhere Anzahl erlaubter Wiederholungen steigert die Zuverlässigkeit der Übertragung bei nur geringfügig höherem Energieverbrauch.
- Leistungsfähige Kodes erfordern die dauerhafte Übertragung langer Bitsequenzen. Dies führt zu einem erhöhten Energieverbrauch und einer höheren Kollisionswahrscheinlichkeit. Daher ist der Einsatz 'guter' Kodes nur in einem schmalen SNR-Bereich sinnvoll, nämlich genau dort, wo der Kode Mehrfachwiederholungen verhindert.

Abbildung 3 zeigt die Aufteilung der Gesamtenergie in die Bestandteile für das Senden (TX), das Empfangen (RX) und im Sleep-Modus. Dabei wurde angenommen, dass der Sensor sich immer im Sleep-Modus befindet, wenn er nicht sendet oder empfängt. Zwei Szenarien aus Abbildung 2 wurden für zwei verschiedene Wahrscheinlichkeiten einer erfolgreichen Übertragung (0.9 bzw. 0.99) betrachtet.

- Der Empfangmodus benötigt in diesem Fall die meiste Energie. Da der Energieverbrauch im Sendemodus allerdings stark von der Sendeleistung und dem Wirkungsgrad des PA abhängig ist, sind Szenarien, in denen die Sendeenergie die Empfangsenergie übersteigt, durchaus realistisch.
- Da Sensoren die meiste Zeit im Sleep-Modus verbringen (sollten), ist auch Anteil der in diesem Zustand benötigten Energie hoch.
- Abhängig vom spezifischen Szenario übersteigt die Energie im Sleep-Modus die des TX-Modus und umgekehrt. Damit ist der Einfluß von Protokoll/Fehlerschutz auf den Energieverbrauch deutlich gezeigt. Auch Parameter der Bitübertragungsschicht, wie z.B. das Modulationsverfahren, würden den Verbrauch der Einzelkomponenten beeinflussen. Dies zeigt noch einmal die Bedeutung einer schichtenübergreifenden Betrachtung bei der Energieoptimierung.

Der Verbrauch im TX/RX-Mode kann dadurch reduziert werden, indem die Daten mit einer höheren Datenrate übertragen werden. Dies führt dazu, dass der Transceiver weniger Zeit im TX/RX-Modus verbringt.
Aktuelle Arbeiten beschäftigen sich u.a. mit der Übertragung der gewonnen Erkenntnisse auf multi-hop Netzwerke.

Literatur

[1] W.R. Heinzelmann, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," *International Conference on System Sciences*, Jan. 2000.

[2] V. Srinivasan, P. Nuggehalli, and R. Rao, "Design of optimal energy aware protocols for wireless sensor networks," *Proc. of the IEEE Vehicular Technology Conference*, vol. 4, pp. 2494–2498, spring 2001.

[3] R. Min et al., "Energy-centric enabling technolgies for wireless sensor networks," *IEEE Commun. Mag.*, vol. 9, pp. 28–39, Aug. 2002.

[4] V. Raghunathan, C. Schurgers, S. Park, and M.B. Srivastava, "Energy-aware wireless microsensor networks," *IEEE Signal Processing Magazine*, vol. 19, pp. 40–50, Mar. 2002.

[5] N. Abramson, "The ALOHA system-another alternative for computer communications," *AFIPS Conference Proceedings*, vol. 36, pp. 295–298, 1970.

[6] E. Callaway et al., "Home networking with IEEE 802.15.4: A developing standard for low-rate wireless personal area networks," *IEEE Commun. Mag.*, pp. 70–77, aug 2002.

Technische Universität Dresden

Betrachtungen zum Energieverbrauch von Sensornetzknoten

Katja Schwieger
Vodafone Stiftungslehrstuhl
Mobile Nachrichtensysteme
Email: schwieg@ifn.et.tu-dresden.de

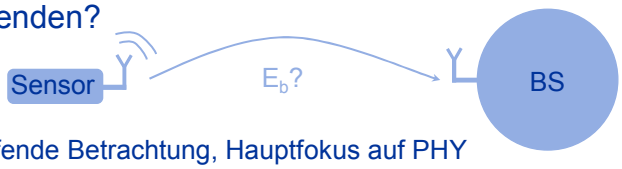
ITG-Fachgespräch
“Drahtlose Sensornetzwerke”
TU Berlin
10./11. Juli 2003

Technische Universität Dresden

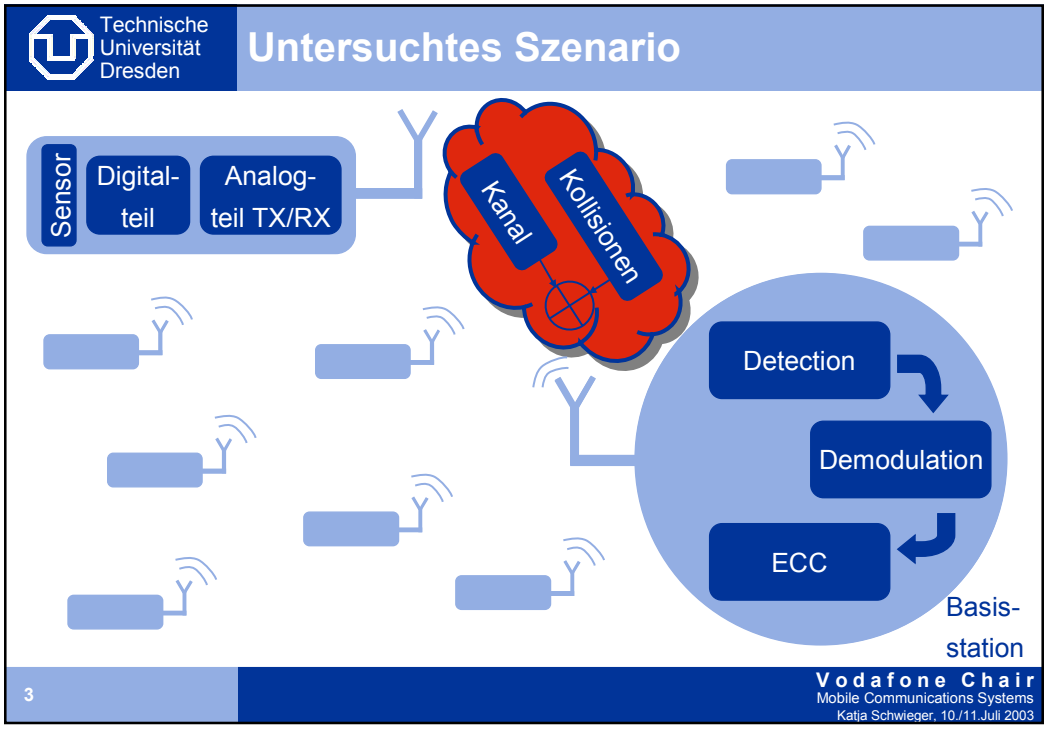
Ziel

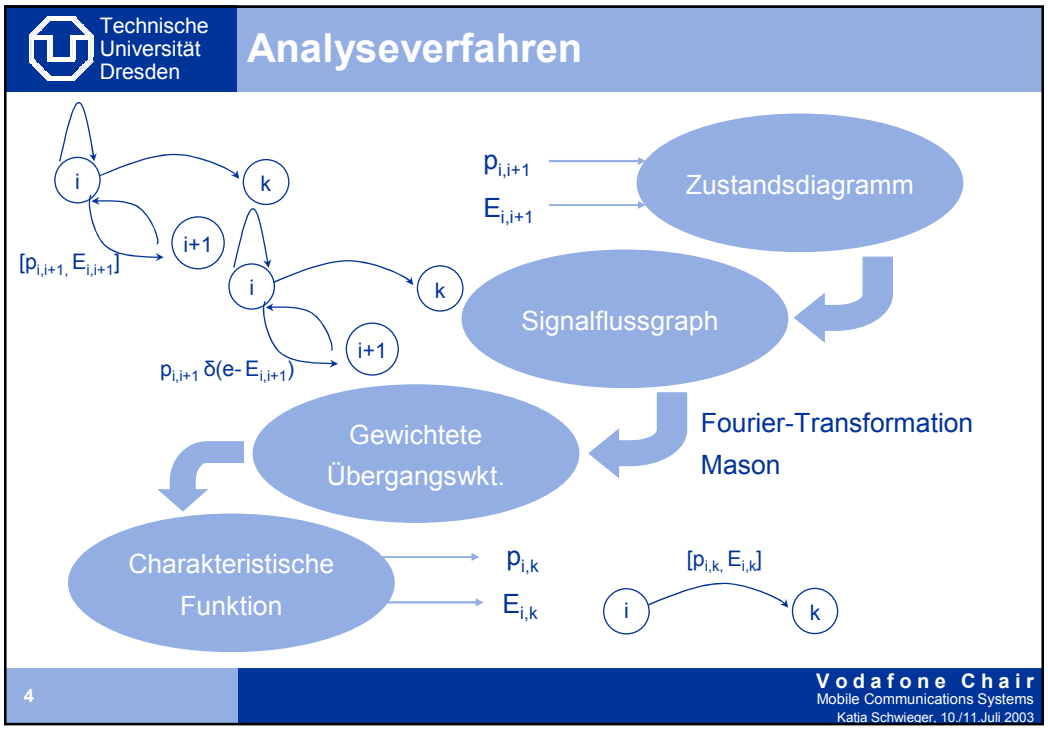
Ermittlung des des Energieverbrauches eines Netzknotens

d.h. wieviel Energie wird benötigt, um ein Datenbit vom Sensor zur Basisstation zu senden?



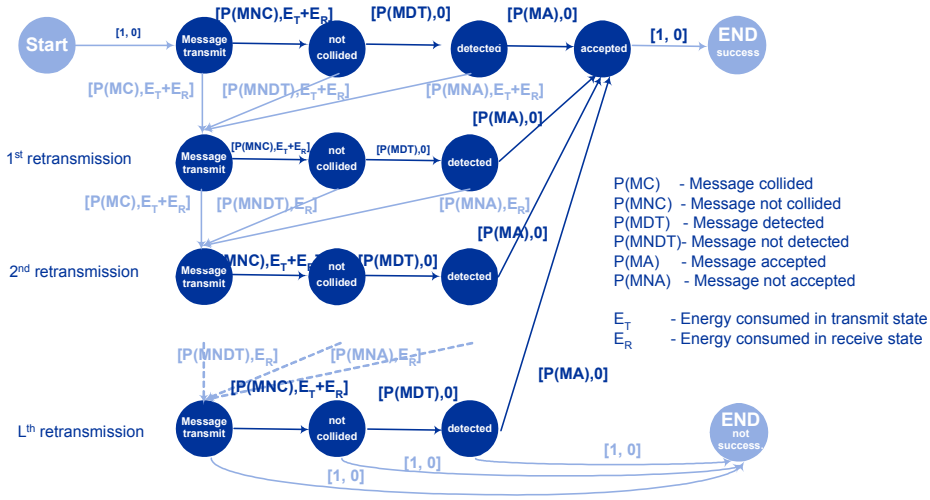
- Schichtenübergreifende Betrachtung, Hauptfokus auf PHY
- Einbeziehung vieler Parameter
- Zuverlässigkeit
- Verteilung des Energieverbrauchs
- Zunächst Betrachtung eines single-link


Technische Universität Dresden
Untersuchtes Szenario
Sensor
Digital-teil
Analog-teil TX/RX
Kanal
Kollisionen
Detection
Demodulation
ECC
Basis-station
3
Vodafone Chair
Mobile Communications Systems
Katja Schwieger, 10./11.Juli 2003


Technische Universität Dresden
Analyseverfahren
i
k
i+1
[p_i,i+1, E_i,i+1]
p_i,i+1 δ(e- E_i,i+1)
p_i,i+1
E_i,i+1
Zustandsdiagramm
Signalflussgraph
Gewichtete Übergangswkt.
Fourier-Transformation
Mason
Charakteristische Funktion
p_i,k
E_i,k
[p_i,k, E_i,k]
4
Vodafone Chair
Mobile Communications Systems
Katja Schwieger, 10./11.Juli 2003

Technische Universität Dresden

Zustandsdiagramm



Technische Universität Dresden

Übergangswahrscheinlichkeiten

$$p_1 = p(MNC) = e^{-\lambda_{net} T}$$

$$p_2 = p(MDT) = \int_{thresh}^{\infty} p_Y(y)dy$$

$$\lambda T = \lambda_{net} T \sum_{l=0}^{L} (1 - p_1 p_2 p_3)^l$$

$p_Y(y)$: non – central χ^2 distribution

$$p_3 = p(MA) = \sum_{i=1}^{m} p_{err}^i (1 - p_{err})^{N-i} \binom{N}{i}$$

$$p_{err} = 0.5e^{-\frac{E_b}{2N_0}}$$

λ_{net}	Gesamtverkehr im Netzwerk
λ	Originalverkehr im Netzwerk
T	Dauer der Übertragung
m	Anz. der korrigierbaren Fehler
N	Paketlänge

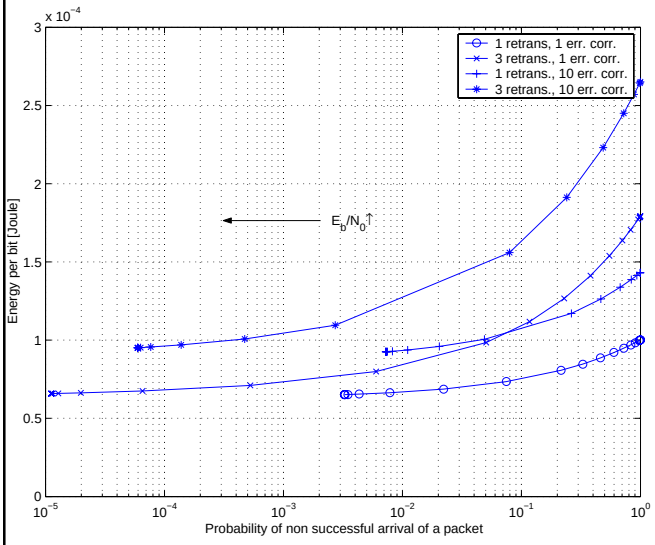
Technische Universität Dresden

Parameter

- PHY:
  Leistungsverbrauch der Komponenten
  Datenrate: 2.4 kbps
  Modulationsverfahren: 2-FSK (nicht-koherent)
  Kanal: AWGN
  Trägerfrequenz: 433 MHz
- MAC:
  Kanalzugriff: slotted ALOHA
  Fehlerschutzkode
- Netzwerk
  Sensorenanzahl, Sendehäufigkeit, Paketlänge, ARQ

Technische Universität Dresden

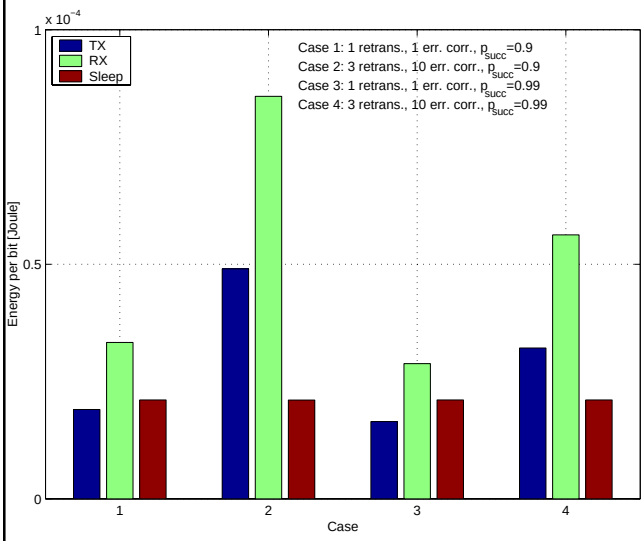
Energieverbrauch vs. Zuverlässigkeit



- ARQ: Geringe Anzahl an Wiederholungen erhöht die Zuverlässigkeit bei geringem energetischen Mehraufwand
- Guter Fehlerschutz benötigt viel Energie
- Hohe Zuverlässigkeit bei hohem E_b/N_0

Technische Universität Dresden

Aufteilung des Energieverbrauches



- Alle Modi verbrauchen signifikanten Anteil an Ressourcen
- Genaue Verteilung stark von Hardware abhängig

Technische Universität Dresden

Ausblick

- Untersuchung des Einflusses weiterer Parameter:
  - Modulationsverfahren
  - Transceiverkomplexität
  - Kanalzugriffsverfahren
- Erweiterung auf multi-hop Protokolle
- Einsatz von UWB für Sensornetzwerke?

Time and Location in Sensor Networks

Kay Römer
Institute for Pervasive Computing
ETH Zurich
roemer@inf.ethz.ch

Abstract

Due to the close coupling of sensor networks to the real world, physical time and location play an important role in many sensor network applications. We explain why this is true and discuss several issues with time synchronization and sensor node localization for sensor networks.

1 Introduction

Recent advances in wireless communication and micro system technology allow the construction of so-called sensor nodes. Such sensor nodes combine means for sensing environmental parameters, processors, wireless communication capabilities, and autonomous power supply in a single tiny device. Large and dense networks of these untethered devices can then be deployed unobtrusively in the physical environment in order to monitor a wide variety of real-world phenomena with unprecedented quality and scale while only marginally disturbing the observed physical processes. Examples include monitoring the behavior of animals in their natural habitats, monitoring the spreading of environmental pollutions in air and water, and monitoring seismic activity and its influence on the structural integrity of buildings.

These exemplary applications demonstrate one important property of sensor networks: their inherent and close integration with the real world, with data about the physical environment being captured and processed automatically, online, and in real time. This has a number of conceptual and technical implications. One such implication is that physical time and location play a crucial role in sensor networks. To understand why this is true, let us consider the basic operation of a sensor network. The functionality of individual sensor nodes is rather simple, they typically measure environmental parameters (e.g., temperature, light intensity) at regular sampling intervals and apply certain filters to the obtained time series of sensor readings to detect "interesting" environmental conditions, emitting a so-called sensor event which describes the detected situation (e.g., the proximity of an observed object). In order to accomplish more complex tasks (e.g., estimating the velocity of a moving object), sensor events obtained from various nodes throughout the network have to be merged in a process called data fusion.

Physical time and location are of importance here for various reasons. Firstly, applications are often interested in time and location of occurrence of a sensor event. Secondly, filters for selecting "interesting" events often contain spatial and temporal constraints. Thirdly, time and location are often a crucial foundation for performing data fusion. Consider for example the estimation of the velocity of a moving object, which can be accomplished by considering the distance in space and time of two "object detection" sensor events obtained from distinct nodes of the sensor network.

For these purposes, nodes of the sensor network have to share a common reference system in time and space, requiring adequate means for time synchronization and node localization. While both of the latter have been examined before in various research contexts, the characteristics and requirements of sensor networks necessitate new solutions. In the following, we will briefly review these characteristics and requirements, before discussing more specific issues with time synchronization and node localization.

2 Sensor Network Characteristics

Individual sensor nodes should by very small (few cubic millimeters), long-living (months to years), cheap, and robust to environmental influences. The small size severely limits the onboard resources of a sensor node (energy, communication bandwidth, computing power, memory). To guarantee longevity despite the limited available energy, all hardware and software components must be consequently optimized for *energy efficiency*. Sensor networks are highly *dynamic*: sensor nodes die due to depleted batteries or harmful environmental influences, new nodes are added to replace failed ones, some nodes are mobile due to environmental factors like wind and water. Despite these dynamics, sensor networks should perform their task in a *robust* way. Due to their potential deployment in remote, inaccessible, or unexploited regions, sensor nodes must operate without excessive external infrastructure – forming so-called ad hoc networks. Many thousands of sensors may have to be deployed for a given task, an individual sensor's small effective range relative to a large area of interest makes this a requirement. Therefore, *scalability* is another critical factor in the design of the system. The expected scale makes manual configuration and management of individual sensor nodes impossible, thus sensor networks should be *self-configuring* and *self-managing*.

3 Time Synchronization

Energy, size, and cost constraints typically preclude equipping sensor nodes with receivers for time infrastructure like GPS [4] or DCF77 [15]. Also, logical time [5] is not sufficient, since it only captures causal relationships between "in system" events, defined by message exchanges between event-generating processes. In contrast, phenomena sensed by sensor nodes are triggered by external physical events which are not defined by in-system message exchanges; physical time must be used to relate events in the physical world.

Time synchronization services for traditional distributed systems like NTP [6] are typically based upon a manually configured hierarchy of network nodes. At the top of the hierarchy are one or more so-called master nodes – canonical sources of time which are synchronized to each other via some out-of-band mechanism such as GPS. Nodes further down in the hierarchy are synchronized to this global time scale by evaluating "time beacons" received from their immediate parent(s). Such beacon messages are frequently sent by a network node to its child nodes, containing the current clock-time of the parent at the time of message generation.

There are various problems with such an approach in the context of sensor networks. As noted above, equipping master nodes with infrastructure such as GPS receivers is typically not an option. In the case of one master (where no external infrastructure for out-of-band synchronization is required), synchronization paths tend to be very long due to the expected scale of sensor networks. This may lead to poor synchronization of nodes far away from the master node. Even worse, nodes which are close to each other, but are far away from the synchronization master, may experience a large synchronization error with respect to each other due to using different synchronization paths to the master with different synchronization quality. This can be a major problem, since colocated nodes tend to require accurate synchronization in order to correlate local sensor events.

Moreover, synchronization schemes like NTP are not optimized for energy efficiency. For example, the CPU is used continuously to perform frequency disciplining of the oscillator by adding small increments to the system clock. In addition, synchronization beacons are frequently exchanged, which also requires constantly "listening" to the network for such beacons. However, with low-power radios used in sensor networks, listening to, sending to, and receiving from the network all require significant amounts of energy. Also, the CPU may not be available if the processor is powered down to save energy.

The manually and statically configured synchronization topology used by NTP is not compatible with the network dynamics in sensor networks. The frequently changing network topology precludes static configuration, the unattended operation of sensor networks precludes manual configuration of individual nodes. Moreover, sensor networks are likely to be temporary partitioned due to node failures or environmental obstructions. Clocks in different partitions are poorly synchronized, which may lead to difficulties when trying to temporally correlate sensor events originating from different partitions after a rejoin of the partitions.

Many of the above problems can be solved by rethinking various aspects of a time synchronization service [3]. Energy efficiency, for example, can be significantly improved by exploiting certain characteristics of sensor network applications. Since sensor networks are typically triggered by physical events, sensor network activity is rather bursty than continuous and rather local than global. This leads to a situation, where synchronized clocks are only required occasionally and only for certain subsets of nodes. Also, the required synchronization accuracy heavily depends on the application, ranging from microseconds (e.g., for acoustic ranging with cm accuracy) to milliseconds or even seconds (e.g., for ordering infrequent events by time of occurence). One possible way to exploit these characteristics is called post-facto synchronization. There, unsynchronized clocks are used to timestamp sensor events. Only when two timestamps have to be compared by the application, they are reconciled to a common time scale.

The problems related to long synchronization paths with varying quality can be avoided by no longer trying to force all clocks of the system to adhere to a global time scale. Instead, local time scales with limited scopes should be established, with timestamps being transformed between scales when crossing a time scale boundary.

In [7] we present a synchronization scheme which adheres to the above principles. There, the unsynchronized clock of each node defines its own local time scale. Timestamps are generated according to this scale. Whenever a timestamp is sent to another node inside a message, a simple computation is used to transform the time stamp to the receiver's time scale. Synchronization can be piggybacked to existing message exchanges, thus keeping the energy overhead for synchronization to a minimum. Also synchronization works across (temporary) network paritions.

4 Node Localization

As with time synchronization, energy, size, and cost constraints typically preclude equipping sensor nodes with receivers for localization infrastructures like GPS. In extreme cases such as Smart Dust [13], it might not even be possible to equip sensor nodes with transceivers for radio waves or ultra sound due to the tiny size and energy budget of Smart Dust nodes. Hence, traditional ranging approaches such as ones based on time of flight of ultrasound signals or received radio signal strength might render unusable in the context of sensor networks.

Many localization systems such as [1, 12] depend on an extensive hardware infrastructure. Localization systems based on trilateration, for example, require many spatially distributed and well-placed infrastructure components in order to achieve high accuracy. For various reasons, this is not an adequate solution for sensor networks. Firstly, this contradicts the ad hoc nature of sensor networks, where nodes may have to be deployed in remote, inaccessible, or unexploited regions. Secondly, sensor nodes often need to know their own location, for example to filter sensor data based on spatial constraints as mentioned earlier in the paper. This may lead to a situation, where many nodes of the network regularly poll the infrastructure for their respective current location, leading to bad scalability of the system. For similar reasons, localization approaches requiring centralized computation such as [2, 11] do not scale well to large networks.

To overcome the limitations of infrastructure-based approaches, various schemes for ad hoc localization have been devised (e.g., [9, 10]). They are typically based on the assumption that few nodes of the network – so-called anchor nodes – know their exact location via some out-of-band mechanism. Other nodes derive their location by, for example, multilateration based on the distances to three or more neighbors with known locations. By iterating this process, all nodes of the network should eventually end up with three or more neighbors with known locations in order to be able to estimate their own location. To avoid accumulating errors inherent to such iterative approaches, many schemes calculate initial location estimates in a first round and iteratively improve these estimates in a second round. However, there are several problems with these approaches. Firstly, good location estimates are only obtained if each node has many neighbors, i.e., if the network is dense. But even then, nodes at the edges of the network tend to end up with poor estimates since they have fewer neighbors. Secondly, the iterative nature of many of the algorithms typically implies a high message overhead, leading to poor energy efficiency.

An important overhead involved in setting up a localization system is node calibration in order to enforce a correct mapping of sensor readings to location estimates

[14]. In systems based on radio signal strength, for example, the received signal strength is mapped to a range estimate. Variations in transmit power and frequency among the nodes can cause significant inaccuracies in the range estimates when used without calibration. Since the cheap low-power hardware used in sensor nodes typically introduces a high variability between nodes, sensor nodes have to be individually calibrated. This, however, may not be feasible in large sensor networks.

Some simple design principles can help solving the above problems. Localized location computation, where nodes autonomously estimate their location without consulting an infrastructure or relying on centralized computations, can help achieve better scalability. As with time synchronization, exploiting certain application characteristics can help improve energy efficiency. The required localization accuracy, for example, heavily depends on the application. In order to track the location of a moving object, for example, a localization accuracy in the order of the size of the tracked object is often sufficient. The calibration problem can be reduced by using differential measurements, where constant offsets cancel out due to using the difference between two measurements that use the same signal path.

In [8] we present a localization system suitable for large networks of tiny sensor nodes. This system consists of a single infrastructure device, which emits certain laser light patterns. By observing these patterns, sensor nodes can autonomously estimate their location with high accuracy. Since sensor nodes only passively observe light flashes, this system is very energy efficient on the side of the sensor nodes. Moreover, optical receivers consume only little power and can be made small enough to fit in a volume of few cubic millimeters. Since sensor nodes do not need to interact with other nodes in order to estimate their location, the system scales to very large networks. Also, sensor node calibration is not necessary due to using differential measurements.

References

[1] N. Bulusu, J. Heideman, and D. Estrin. GPS-less Low Cost Outdoor Localization for Very Small Devices. *IEEE Personal Communications*, 7(5):28–34, October 2000.

[2] L. Doherty, K. S. J. Pister, and L. E. Ghaoui. Convex Position Estimation in Wireless Sensor Networks. In *Infocom 2001*, Anchorage, Alaska, April 2001.

[3] J. Elson and K. Römer. Wireless Sensor Networks: A New Regime for Time Synchronization. *ACM SIGCOMM Computer Communication Review (CCR)*, 33(1):149–154, January 2003.

[4] B. Hofmann-Wellenhof, H. Lichtenegger, and J. Collins. *Global Positioning System: Theory and Practice, 4th Edition*. Springer-Verlag, 1997.

[5] L. Lamport. Time, Clocks, and the Ordering of Events in a Distributed System. *Communications of the ACM*, 21(4):558–565, July 1978.

[6] D. L. Mills. Improved algorithms for synchronizing computer network clocks. In *Conference on Communication Architectures (ACM SIGCOMM'94)*, London, UK, August 1994. ACM.

[7] K. Römer. Time Synchronization in Ad Hoc Networks. In *MobiHoc 2001*, Long Beach, USA, October 2001.

[8] K. Römer. The Lighthouse Location System for Smart Dust. In *MobiSys 2003*, San Franscisco, USA, May 2003.

[9] C. Savarese, J. M. Rabaey, and K. Langendoen. Robust Positioning Algorithms for Distributed Ad-Hoc Wireless Sensor Networks. In *USENIX Annual Technical Conference*, Monterey, USA, June 2002.

[10] A. Savvides, C. C. Han, and M. Srivastava. Dynamic Fine-Grained Localization in Ad-Hoc Networks of Sensors. In *Mobicom 2001*, Rome, Italy, July 2001.

[11] Y. Shang, W. Ruml, Y. Zhang, and M. Fromherz. Localization from Mere Connectivity. In *ACM MobiHoc 2003*, Annapolis, USA, June 2003.

[12] R. Want, A. Hopper, V. Falcao, and J. Gibbons. The Active Badge Location System. *ACM Transactions on Information Systems*, 10(1):91–102, 1992.

[13] B. Warneke, M. Last, B. Leibowitz, and K. S. J. Pister. Smart Dust: Communicating with a Cubic-Millimeter Computer. *IEEE Computer Magazine*, 34(1):44–51, January 2001.

[14] K. Whitehouse and D. Culler. Calibration as Parameter Estimation in Sensor Networks. In *Workshop on Wireless Sensor Networks and Applications (WSNA) 02*, Atlanta, USA, September 2002.

[15] DCF77 Radio Time Signal. www.dcf77.de.

Clustering in Wireless Sensor Networks: Challenges and Applications

Petia Todorova

FhG-FOKUS

todorova@fokus.fraunhofer.de

Extended Abstract

1. Wireless Sensor Networks – A New Generation of Ad - hoc Networks

Ad-hoc wireless networks of deeply embedded devices such as micro-sensors have emerged as one of the key growth areas for wireless networking and computing technologies. A wireless network of the type investigated here refers to a group of sensors, or nodes, linked by a wireless medium to perform distributed sensing tasks. Connections between nodes may be established using such media as infrared devices or radios. Although many protocols and algorithms have been proposed for traditional wireless ad-hoc networks, they are not well suited to the unique features and application requirements of sensor networks. The main differences between sensor networks and ad-hoc networks can be summarized as follows:

- Sensor nodes are limited in power, computational capacity, and memory;
- Sensor nodes are prone to failure;
- The topology of the sensor network changes very frequently;
- Sensor notes use a broadcast/multicast communication paradigm;
- Sensor nodes are densely deployed.

The position of sensor nodes need not be engineered or predetermined, thus allowing random deployment even in inaccessible terrains. This means that sensor networks algorithms and protocols must possess self-organizing capabilities. Hence, network self-assembly and continuous network self-organization during the lifetime of the network in an efficient, reliable, and scalable manner are crucial for the successful deployment and operation of such networks.

Clustering was proposed in large-scale mobile ad-hoc networks (MANET) as a means of achieving scalability through a hierarchical approach. Considering energy as an optimization parameter in sensor networks clustering becomes a very important optimization problem. At the same time it ensures high scalability of the network through its hierarchical approach.

2. Clustering – Challenges

Clustering is defined as the grouping of similar objects or the process of finding a natural association among some specific objects or data. Group communication is natural for sensor networks for the following reasons: some tasks can not be done by a single sensor, to avoid redundant deployment, for increasing reliability and optimize energy consumption. Since the way in which nodes are grouped defines how a clustering algorithm behaves, the necessity to define the characteristics of the grouping scheme: membership size, group leader (clusterhead), group overlap, size of the group (cluster depth), depending on the application, is the first step towards clusters establishment in the network.

2.1 Clustering – A Tool for Energy Optimization

Communication is usually the main source of energy dissipation in sensor networks, which greatly depends on the distance between source and destination. We draw our attention to the main directions in the area.

In **LEACH** (Low Energy Adaptive Clustering Hierarchy) a clustering formation protocol randomly selects sensor nodes as clusterheads, so the high energy dissipation is spreaded to all nodes [1]. Also clusterheads send directly to the end user, saving energy this way, for large networks the protocol poses limitation because of the fact that some clusterheads may be deployed far from the end user.

Energy optimization based on **balanced clustering** [2] minimizes the total distance between sensors and master (clusterhead), assuming exactly one master per cluster. At the same time upper/lower bound on the size of each cluster is defined. The algorithm performs very well with 24 nodes, clusterhead handling 6 sensor nodes. To proof the concept one has to increase the number of nodes in the network. Banerjee and Khuller [3] extend the idea above proposing a new algorithm for construction **overlapping clusters of bounded size**.
The algorithms based on the rational that low-diameter decompositions lead to low end-to-end delay reducing energy consumption have restricted mostly to low density networks. In densely connected networks these algorithms can allow the cluster size to be violated arbitrarily.
A novel approach, solving partly the problem is based on the concept of **budget allocation** [4]. That approach significantly reduces the number of messages exchanged and allows the cluster to grow based on local decisions rather than involving the initiator, producing clusters of bounded size.
A scalable energy-efficient **training protocol** for nodes that are initially anonymous, asynchronous and unaware of their location is proposed in [5]. Training protocol partitions the node into clusters and provides a virtual tree for efficient routing from clusters to sink.

2.2 Clustering – A Network Self-assembly Mechanism

The goal of clustering used as network self-assembly mechanism at the link level is to enable distributed formation of a connected wireless backbone and maintain the connectivity as the topology changes due to removal or addition of nodes, etc. Multi-clustering enables the formation of a scalable network topology allowing interconnect clusters in the sensor network. Certain nodes must be member of multiple clusters to ensure network connectivity. In [6] two different cluster formation mechanisms are proposed, assuming dual radio architecture of the nodes. The clusters are formed directly as a consequence of the radio's MAC scheme. They are called the "**Dual Network Clustering**" **(DNC)**, and "**Rendezvous Clustering Algorithm**" **(RCA)**. The DCA has the advantage of being relatively simple to implement. However it lacks configurability to enable certain level of control when determining node membership in a cluster. In the RCA one of the two radios on a node is tuned to a fixed signalling channel in the networks. The relationship *clusterhead – cluster member* is better defined and increases performances.
Also the algorithms provide for a scalable network topology, energy optimization is not taken into account.

3. Open Research Issues and Future Work

There is a need to improve existing protocols or develop new energy efficient protocols addressing network topology changes and role changes, providing for connectivity and scalability of the sensor network. Work in progress is focused on efficient clustering algorithms with network topology changes.

References

[1] Wendi Rabiner Heinzelman, Anantha Chandrakasan, and Hari Balakrishnan , "Energy-Efficient Communication Protocol for Wireless Microsensors Networks", Proceedings of the Hawaii International Conference on System sciences, January 4-7, 2000, Maui, Hawaii
[2] Soheil Ghiasi, Ankur Srivastava, Xiaojian Yang, and Majid Sarrafzadeh, "Optimal Energy Aware Clustering in Sensor Networks", Sensors 2002, 2 258-269.
[3] S. Banerjee and S. Khuller, "A Clustering Scheme for Hierarchical Control in Multi-hop Wireless networks", Proc. IEEE INFOCOM 2001, Anchorage, AK, USA, April 2001.
[4] Rajesh Krishnan and David Starobinski, "Message-Efficient Self-Organization of Wireless Sensor Networks, http://people.bu.edu/staro/wcnc-rajesh.pdf
[5] A. Wadaa, S. Olariu, L. Wilson, and Q. Xu, "On Training a Sensor Network", Proc. of the 3-rd International Workshop on Wireless, Mobile and Ad-hoc Networks, April 2003, Nice.
[6] Katayoun Sohrabi, William Merrill, Jeremy Elson, Lewis Girod, Fredric Newberg, and William Kaiser, Scalable Self Assembly for Ad Hoc Wireless Sensor networks, http://dsp.jpl.nasa.gov/cas/pull/sohrabi.pdf

Fraunhofer Institute for Open Communication Systems

Clustering in Wireless Sensor Networks:

Challenges and Applications

Fraunhofer Institute for Open Communication Systems

Content

- Introduction
- Clustering – requirement and challenges
- Clustering as a tool for energy optimization
- Clustering as a networks self - assembly mechanism
- Concluding remarks
- Open research issues

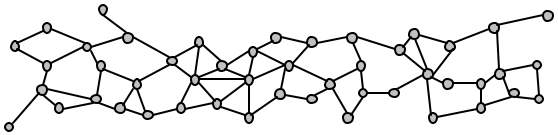
Fraunhofer Institute for Open Communication Systems

Introduction

- Sensor nodes are limited in power, computational capacity, and memory
- Sensor nodes are prone to failure
- The topology of the networks changes very frequently
- Sensor nodes use a broadcast/multicast paradigm
- Sensor nodes are densely deployed

Fraunhofer Institute for Open Communication Systems

Clustering



- To provide infrastructure for location and time-synchronization procedures
- Optimize energy consumption
- For increasing reliability and scalability
- Support data--fusion
- Autonomous set up of addressing and (hierarchical) routing in large networks

Clustering – requirements

- Develop energy-efficient solutions
- Distributed self-assembly (organizing) algorithms
- Adapt to topology changes
- Divide the network in local sub-groups
- Provide for network scalability

Clustering – requirements

- Define groups with
  - Common capability (e.g. localization)
  - Common task (e.g. routing)
  - Common interest (application dependent)
- Assign each sensor node to a specific master node (cluster-head)
- Define the characteristics of the clustering scheme: membership, cluster-head, cluster size

Clustering – organization

- Topological clustering
  - Grouping is provided by proximity
  - Provides infrastructure for hierarchical routing
  - Reduces routing overhead
- Information clustering
  - Defined by data-centric approach
  - Data fusion – important task
  - Connectivity not a concern

Clustering – a tool for energy optimization

- LEACH (Low Energy Adaptive Clustering Hierarchy)
  - Includes randomized rotation of cluster-heads position
  - Energy dissipation is spread to all nodes
  - Advertisement, cluster set-up, TDMA creation, data transmission
  - Extended to form hierarchical clusters

Fraunhofer Institute for Open Communication Systems

Clustering – a tool for energy optimization

- Balanced clustering algorithm
  - Upper/lower bound on the size of each cluster is defined
  - Minimizes the diameter between sensors and cluster-head – reduces the communication overhead
  - Performs very well with 24 nodes, clustering handling 6 sensor nodes

Fraunhofer Institute for Open Communication Systems

Clustering – a tool for energy optimization

- Budget allocation algorithm RAPID
  - Sensor nodes allocate local *growth budgets* to neighbors
  - The initiator is provided a budget of ***B,*** of which it accounts for itself and distributes ***B - 1*** among its neighbors via messages
  - The messages propagate until they reach a stage where the budget is exhausted producing clusters bounded by **B**
  - The approach significantly reduces the number of messages exchanged

Fraunhofer Institute for Open Communication Systems

Clustering – a tool for energy optimization

- Training protocol
  - Imposes a coordinate system in such a way that the nodes in a cluster have the same coordinates
  - The sensor network is divided into equiangular wedges
  - The wedges are divided into sectors by means of coronas centered at the sink
  - The mapping between clusters and sectors is one-to-one
  - Scalable, lightweight, and energy efficient

Fraunhofer Institute for Open Communication Systems

Clustering – a network self-assembly mechanism

- Dual Network Clustering
  - The radios of a node are each tuned to a specific, well-known fixed channel
  - The first radio that wakes up and finds a quiet channel becomes the cluster-head of the channel
  - The radios that arrive to that channel at later times become members of the cluster

Fraunhofer Institute for Open Communication Systems

Clustering – a network self-assembly mechanism

- Rendezvous Clustering
  - Separate signalling channel (R - channel) for exchange of connectivity information and decide about roles
  - The cluster-head uses one of its radios to surf the R - channel, while the other radio is used to talk to the cluster members
  - After getting connected, the cluster member uses its radios to talk to two different clusters

Fraunhofer Institute for Open Communication Systems

Concluding remarks

- Clusters should be bounded and moderately sized
  - Small clusters are inefficient
  - Large clusters overburden the cluster head
- Initiator rotation could allow for recovery in the event of failed nodes
- For large networks hierarchical clustering could save a significant amount of energy
- Multi-cluster mechanisms increase networks scalability

Fraunhofer Institute for Open Communication Systems

Open research issues

- Improve existing protocols addressing network technology changes and role changes
- Develop cluster maintenance protocols
- Flexible control cluster size
- Develop attribute -based clustering

Consistent Update Diffusion in Sensornetworks

Christian Becker, Jörg Hähner, Pedro José Marrón
University of Stuttgart
Institut für Parallele und Verteilte Systeme (IPVS)
Universitätsstr. 38
70569 Stuttgart
{Becker,Haehner,Marron}@informatik.uni-stuttgart.de

Abstract
Sensor networks are envisioned in scenarios where network infrastructures are not available either due to damage, e.g. in rescue missions, or by design, e.g. surveillance of bird populations or forest fire detection. When distinct changes in the environment are sensed, sensor nodes propagate updates of object states into the sensor network. In this extended abstract we introduce a new consistency concept that takes the global ordering of object state changes into account. Sensors propagate state changes to mobile nodes, each of which keeps local copies of object states. Our novel consistency concept ensures that applications will only read the most recent value of an object.

Motivation
The observation of real-world objects is a typical task sensor networks are normally used for. Whenever a change in the real world occurs, updates propagate state changes into the sensor network towards sinks that process these updates and react accordingly. Apart from static environments, sensor nodes could be deployed in areas where a combination of mobile and sensor nodes is given. For example, think of buildings in earthquake endangered areas. If sensors are deployed in the building, they can be activated in case of an earthquake and provide information about the environment, such as life-signs, gas-detection, etc. Members of a rescue mission can pick up this information as they pass through the disaster area. In order to react accordingly, a complete picture of the missions is required. Sensors propagate their information to a mobile device of a mission member which further disseminates the information to other devices as soon as they are in radio range. Clearly, if information about an object is delivered by more than one sensor node, the real time ordering of the update events is crucial in order to deliver a consistent view. Due to network partitions in the mobile ad hoc network and the possible network partitions between sensor devices, the global ordering of update events is hard to achieve. We propose a new consistency concept called *update linearizability*, which ensures that clients read only the most recent state of an object. In extensive simulation experiments we were able to show that the implementation of an algorithm ensuring update linearizability requires only little overhead for synchronization. Furthermore, the data available at the replicated copies is almost up-to-date even in networks with high load.
In the following, we will briefly describe our system model, define the consistency concept, and sketch the algorithm.

System Model
We consider two kind of nodes: sensor and database nodes. *Sensor nodes*, also called observers, are mobile or stationary and sense information about their environment. If they detect a state change they emit an update request for the sensed object, say x, along with the sensed state. *Database nodes* are mobile and keep a local database with copies of the state of all objects. Whenever a database node receives an update request for an object, say x, it has to check whether the local stored state information is older than the received one and accept or reject the update request accordingly. Moreover, database nodes propagate update requests only if they are newer than their local state, so as to keep the database instances of other database nodes up to date.

Consistency Concept
Due to the lack of global time in distributed systems, the real-time order of observation events occurring at different observers can be captured only with limited accuracy. Therefore, we introduce the *occurred-before* relationship for update requests, which is a relaxation of real-time ordering.

Definition 1 (occurred-before): Let *u* and *u'* be two update requests. Then *u occurred-before* (<) *u'* iff *observation_time(u') - observation_time(u)* > δ, where δ > 0 and *observation-time(u)* denotes the real time at which the observation leading to the generation of *u* occurred.
Parameter δ is a system parameter that defines how precisely the real-time ordering of observation events is captured by the underlying system. This parameter is important for applications because it defines the minimum temporal distance of any pair of observation events needed to determine their correct real-time ordering. If neither $u < u'$ nor $u' < u$, then *u* and *u'* are said to be *concurrent*, denoted as $u \parallel u'$. For concurrent update requests it cannot be guaranteed that correct real-time ordering is captured.
Based on the *occurred-before* relation, *update linearizability* can be now defined as a weaker consistency model than *linearizability*, where the ordering of both update and read operations is consistent with the real times at which the operations occurred in the actual execution. Update-linearizability is much easier to achieve than linearizability and is sufficient for many applications that require current information about real world objects. Formally, we have:

Definition 2 (update-linearizability): An execution of the read and update operations issued by clients and observers is said to be *update-linearizable* if there exists some serialization *S* of this execution that satisfies the following conditions:
(C1) All *read operations* of a single client on a single object in *S* are ordered according to the program order of the client
(C2) For each object *x* and each pair of update requests *u[x]* and *u'[x]* on *x* in *S*:
u'[x] is a (direct or indirect) successor of *u[x]* in *S* iff $u[x] < u'[x]$ or $u[x] \parallel u'[x]$
(C3) Each object o in the DB *S* meets the specification of a single copy of *o*

Definition 2 captures the idea that for a given execution of operations, there exists a serialization of this execution against a single logical image of the DB objects, and each client sees a view of the (physical) objects that is consistent with the logical image. It guarantees that updates are performed only in the order defined by the *occurred-before* relation. Once a client *c* has executed a read operation r_1 that returned the result of an update request u_1 on a specific object *x*, condition C1 guarantees that the next read operation r_2 of *c* on *x* will at least return the same result as r_1 or some result written by an update request u_2, with $u_1 < u_2$. C2 ensures that u_1 *occurred-before* u_2 or both are concurrent, preventing clients from reading older values in their subsequent program order. This means that the result of read operations may be the same *as if* their execution was delayed, yet ensuring that once a value has been read, no older value will be returned in succeeding read operations. Condition C3 ensures that a read operation on *x* returns the result of the preceeding update request on *x* in *S*.

Update Algorithm

The algorithm consists mainly of two protocols: the observer-to-node and node-to-node protocol. The former is defined between observers and nodes to transmit direct update requests from a given observer to a node. The algorithm guarantees that if an update is executed on the node, it is because the update request is newer that the current state of its database regarding the observation. The second protocol defines the execution of update requests between two nodes. In order to decide whether an update should be accepted, each node keeps an ordering graph that allows it to discriminate older updates without the need of a global synchronized clock between nodes. Hence, every node can decide whether or not it stores an update locally based on its ordering graph.

University of Stuttgart
Institute of Parallel and
Distributed Systems (IPVS)
Universitätsstraße 38
D-70569 Stuttgart

Consistent Update Diffusion in Sensor Networks

Christian Becker, Jörg Hähner, Pedro José Marrón

Outline

- Motivation
- System Model
- Consistency Model
- Algorithm

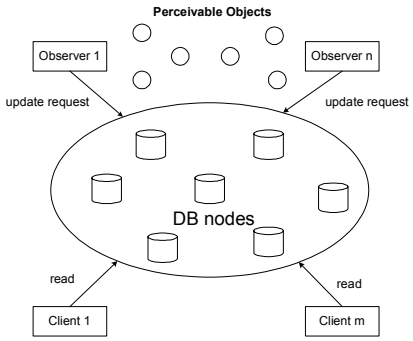
Motivation

- Sensor networks
  - Capture aspects of real world objects
  - Objects may be observed by independent sensor nodes
  - Real-time order important
  - Typically highly distributed
- Many applications potentially need
  - Access to a global state of all aspects captured
  - High availability of state
- Spectrum of approaches
  - Complete replication on user devices → local queries
  - Partial replication
  - No replication

  } → remote queries

System Model

- Perceivable objects
  - Unique identifier
  - State relevant to the application
- Observer nodes
  - Contain sensors that monitor the state of objects in their vicinity
  - Transmit state changes of objects → *update requests*
- Client nodes for users
  - Run user application
  - Maintain a copy of the most recent state of each object
  - Exchange information with each other



Consistency Model: Definition of Ordering

- Real-time ordering only possible with limited accuracy
- Definition: *occurred-before*

  u ***occurred-before*** *(<) u*,* iff

  observation_time(u) – observation_time(u) > delta > 0*
- System parameter *delta*
  - Describes precision of ordering
  - Difference less than *delta* → ***concurrent*** update requests

Consistency Model

- Definition: *update-linearizability*
  - ∃ a **serialization** S for the executions of all clients and observers:
  - All ***read operations*** of a single client on a single object in *S* are ordered according to the program order of the client
  - For each **object *x*** and each **pair of update requests** *u[x]* and *u'[x]* on *x* in *S*:
    u'[x] is a (direct or indirect) successor of *u[x]* in *S* iff
    u[x] < u'[x] or *u[x] || u'[x]*
  - *Each object o* in the DB *S* meets the specification of a single copy of *o*

Algorithm: Basic Ideas

- Assumption: maximum time between state change at object and arrival of data at **first** node is bounded
- Two protocols: observer-to-node and node-to-node
- Ordering graph represents global time ordering
  - Constructed by observer-to-node protocol
  - Merged by node-to-node protocol

Energy-efficient Publish/Subscribe Routing for Wireless Sensor Networks

Vlado Handziski

June 4, 2003

1 Motivation

WSN are tightly coupled with the physical environment. The sole purpose of their existence is to provide information about the environment in which they are. The final aim is to get this observable data efficiently to the interested parties. The content-based publish/subscribe is very suitable for this type of interaction and will simplify the application development by providing to the programmer a high-level construct that is much close to the application domain than the classical UDP socket service, for example.

Another defining characteristics of the sensor networks is their unreliability. It is expected that the sensor nodes will fail/recover or go in sleep mode in order to preserve energy. One way to counter these issues is by over-provisioning the network with nodes (in relation to the minimum number of nodes required for the requested quality of information gathering), thus introducing increased redundancy. This approach poses some interesting challenges to the messaging sublayer in the network. We need a mechanism that will provide an efficient level of indirection that will mask this dynamics from the upper layers, shielding the application from the network volatility. The content-based nature of publish/subscribe can present such a level of indirection. It does not require an additional name resolution step which can substantially reduce the required message traffic, especially in the case of frequent and short-lived exchanges.

In WSNs there are many different types of traffic:

- Data-dissemination: Periodic or Triggered
- Control-information: Neighborhood information, Time synchronization, Localization, Management, Service discovery..

They can benefit from data-centric routing in different degrees. The gain is maximized if the exchange:

1. Needs name resolution (so we save on the name resolution triangle).
2. Can be optimized if one has access to application level data (so we save by in-network processing)

While the naming part of a content-based publish/subscribe defines its expressive power, the routing and the forwarding part determine its efficiency.

In contrast to the subject based Publish/Subscribe systems that can be nicely supported by the current multicast based ad hoc routing protocols, the content-based publish/subscribe requires more sophisticated routing structure. How to create and maintain such a structure in the environment that is specific to the wireless sensor networks is the focus of our interest.

2 Related work

There are several routing approaches proposed for use in distributed publish/subscribe systems:

- "Classical" (IP based) publish/subscribe systems:
  - Simple flooding: Subscriptions are flooded in the network

  - Mapping to IP multicast: In subject based publish/subscribe the targets are directly mapped to the underlying multicast routing groups.
  - Covering routing [1]: Do not forward the subscription if a covering one is already sent to that neighbor.
  - Merge routing [5]: Before forwarding a subscription, try to create a merged subscription that covers the range of the new and the old subscriptions and then forward this single subscription.

- Ad hoc and WSN proposals:
  - Interval routing [4]: Single distribution tree with fixed table lengths, based on the interval routing protocol.
  - Publish/subscribe trees [2]: Actively maintained distribution tree with parent selection heuristics that is based on the additional overhead that the children subscriptions bring.
  - Directed Diffusion [3]

From the above WSN proposals, Directed Diffusion the most visible one. It applies some of the ideas of publish/subscribe in the domain of sensor networks. Very briefly, the protocol mechanics is as follows:

1. Subscribers flood the subscriptions in the form of INTEREST messages. Gradients are formed towards the nodes from which I got the INTEREST. At the end, this creates gradients between each two neighbor nodes.
2. The matching sinks publish EXPLORATORY DATA along these gradients. When the data hits the subscriber, he sends REINFORCEMENT messages that reinforce one / couple of the paths based on some metric. The tree can also be pruned using NEGATIVE REINFORCEMENTS.
3. The normal data is sent just along these reinforced links.
4. The routing structure is maintained by periodic reflooding of INTEREST messages and EXPLORATORY DATA messages.

As we have shown in our Sensys submission, the main problem with this approach is the control overhead. The network is overwhelmed by the flooding of the INTERESTS and EXPLORATORY DATA. When due to synchronization (and in the case of multiple subscribers) this phases coincide the network becomes fully congested for long periods of time.

Noticing this, the original authors in a very recent submission suggested that the EXPLORATORY DATA phase should be skipped and that the real data should be sent directly to the subscriber along single path. For example, send along the gradient to the neighbor that first sent the interest message.

This changes significantly lower the amount of control traffic, on the expense to the robustness of the algorithm: it strengthens the symmetrical channel assumption and it requires that flows are marked with a flow ID, thus losing the "pure local" nature of the algorithm.

3 Publish/Subscribe for WSNs

The benefit of the semantic addressing in WSNs is clear. This "intentional naming" provides the necessary space decoupling and is obviously closer to the sensor network application space as elaborated in section 1.

But what about the other axes of asynchrony ? What is the degree of time and flow decoupling that is necessary in WSNs ? The answer to this question depends on the intended target applications and their nature. We can roughly divide them in two classes: *monitoring* and *event-triggered*.

In the first class are all applications that perform constant monitoring of the environment based on the data collected by the sensor network. The data is potentially and probably subjected to in-network processing on the way from the producers to the subscribers.

In the simplest case, there is a single subscriber (sink). The distribution network is usually in the form of a converge-cast tree routed in the subscriber. Various techniques to increase the reliability of this rather vulnerable structure are relevant here.

Extending this scenario to more subscribers brings additional challenges. The number of sub-

scribers is still going to be substantially smaller than the number of potential producers in the network. The distribution network can now consist from multiple distribution trees routed in the subscribers. There are couple of opportunities for optimization as a result of the availability of application level information: The potential overlap in the subscriptions can be exploited to lower the control overhead (overlapping, covering, merging subscriptions). So building a joint distribution tree is an option. Tree vs. mesh-based approaches should be considered. This routing structure has to efficiently support the desired in-network processing. In this application scenario there is little need for temporal and flow separation.

The (attribute, operation, value) addressing implicitly defines a group of nodes based on the relationship between some property of the nodes and some absolute external value. Each node that has been pre-programmed with the valid attributes and their ranges can determine for it self, without any additional external information whether they belong to the group or not. So they can individually decide if they should start publishing or not.

But there are many applications in WSN where the elementary cooperation group of nodes can not be absolutely defined. The membership of this group is dependent on the relation between some of the characteristics of the node's attributes with the ones of other nodes (for example, we need to talk with the node that has the highest temperature among all the nodes in 2m radius). So there is a need for exchange of information before a node can decide if it belongs to this group or not.

Luckily, most of the time, the membership can be evaluated based on local information, i.e. based on information exchange with the neighbors of the node. From a communication point of view, this can be solved in the flat case with one-hop broadcasts or with caching. In the clustered scenarios, the cluster head can evaluate this relationships and determine the membership functions.

The problem is far more complicated when the defined relationships are not local in their nature, i.e. when the relations are expressed over semantic addresses that do not correspond to a physical cluster. This would require a routing structure substantially different from the one proposed for "simple" monitoring applications.

An efficient routing structure in semantic space will require grouping based on some common semantic properties, so whole branches can be pruned when the queries or the data they carry is not relevant for the answering of the original question. This is some form of indexing. But it introduces two major problems:

1. It creates an overlay. For the semantic properties that are not tightly correlated with radio range, this will result in discrepancy between the communication topology and the most optimal data distribution tree. There is also a problem in the form of the different levels of dynamics. In static networks, the physical topology is going to be rather constant, or at least slow changing, while the optimal semantic routing structure is highly dynamic as it is dependent on the query that is executed at the moment.

2. This overlay structure is also very brittle. Physical node failures, sub/un-subscription will trigger restructuring. How to make the structure more robust ?

4 Towards the solution

Having in mind the open issues discussed in the previous section, what are the types of naming and routing that the suggested publish/subscribe solution should have ?

The proposed improvements can be grouped in two classes: *naming improvements* and *routing/forwarding improvements*.

4.1 Naming improvements

1. It should be based on the principles of content-based networking where the addressing is in the form of event patterns usually specified in the form of the (attribute, operation, value) tuple. This provides sufficient expressing power for large range of applications.

2. This naming should be additionally tailored to sensor networks by including a confidence parameter for example. This parameter can have several uses. The most basic one is for specifying the "extent" of the multicast to the

implicitly defined group. For example: (temperature, >,30, 50%) means that the distribution network should try to deliver the message to about 50% of the nodes that have temperature higher then 30 degrees.

This will provide to the user a tuning knob for matching the trade-off between accuracy and energy-efficiency to the specifics of the given application.

This parameter also provides space for load-balancing in the implicitly defined group, as we can change the sampled members with each successive exchange.

4.2 Routing improvements

The distribution tree should guarantee efficient and accurate distribution of the publications to the interested parties. At this level, the service is of best-effort type. It is left to the upper layers to implement additional services as: reliable delivery, exactly once delivery, temporal order delivery, etc.

As discussed in Section 3, one-fits-all routing solution is very improbable. What we probably need is several optimized solutions for the identified application classes.

In the design of the protocols we have to address but also use for our benefit the peculiar characteristics of the wireless sensor network environment. In particular we intend to use the broadcasting property of the wireless medium and heavily rely on the tight integration between the protocols layers in for increased efficiency.

One of the crucial problems that has to be tackled is the issue of the varying efficiency of the distribution network based on the communication patterns and the level of overlapping and locality of the subscriptions. This is the same problem that overlay networks are facing, but the penalty for the mismatch between the physical topology and the publish/subscribe distribution network is substantially greater in WSNs then in internets.

One possible solution is that in the creation and maintenance of the distribution network, we have to use a flexible cost function that combines both the "semantic" and the "physical" costs. The former one, for example can depend on the level of non-overlapping between the subscriptions, while the later can be casted into terms like number of hops or real distance if location information is available.

The flexibility and the robustness of the network will also depend on its structure. The majority of proposed approaches in the literature are based on trees routed in the subscriber or publisher, depending on the expected ratio between subscriptions and publications. The issue of reusing the existing structure when adding new subscriptions is seldom tackled. Recent publications in the ad hoc networks field, show that the mesh structure is superior (in terms of reliability) to the shared trees approach, at the cost of increased overhead [6]. Whether this is true in the case of sensor networks under the publish/subscribe communication pattern is still unclear and warrants careful deliberation.

Another open issue is the trade-off between building the distribution network completely on-demand and relying just on local (one-hop) info, thus resulting in suboptimal solutions, and the introduction of limited active collection of "semantic" information in the extended neighborhood, that can potentially result in distribution network with much better filtering capabilities.

All of these questions have to be answered before we reach the grail of flexible publish/subscribe middleware for WSNs that is able to act as a usable building block for the plethora of envisioned applications in this space.

5 Methodology

In the pursuing of our goal, we are mainly going to rely on simulation studies. Because of its flexibility, ease of use and good debuging capabilities, we plan to use the OMNET++ discrete event simulator. The functional testing is going to be done with a null MAC, so that the merits of the different approaches on the publish/subscribe level are more easily discernible. The performance testing will proceed with full network stack simulation. At the end, a down scaled protocol can be implemented in the EYES test bed and experimentally verified.

5.1 Models and parameters

- *Distribution network:*

1. Number of nodes
2. Average degree
3. Average connectivity
4. Spatial distribution

- *Subscribers (sinks):*
  1. Number
  2. Spatial distribution
  3. Rate (or parameters of the distribution) of the subscriptions
  4. Overlapping/Locality of the subscriptions
- *Publishers (sources):*
  1. Number
  2. Spatial distribution
  3. Rate (or parameters of the distribution) of the publishes
  4. Rate (or parameters of the distributions) of the advertisements

Most of the above parameters are also traditionally used in evaluation of multicasting protocols for ad hoc networks, so selecting relevant types/values should not pose a big problem. Unfortunately, this can not be said for the type and the parameters of the random distribution that should describe the subscriptions, publishes and advertisements. In the face of lacking solid "field" data from operational sensor networks, we have to resort to some sort of approximations that at least capture the main characteristics of the message exchange. The model used by Huang and Garcia-Molina [2] can be a starting point:

- *Number intervals:* Subscriptions are modeled as intervals on the real axes. They can be closed or open. For example: [20,50], [20,], [,50] etc. The intervals are formed by selecting a central value and length. The length is selected randomly from a normal distribution. The center of the interval can be selected randomly from the interval of possible values. For modeling subscriptions with locality characteristics (which is crucial for testing the gains of the cover/merge mechanisms) one can use the x or y coordinates of the node to influence the selection of the center. This introduces the necessary correlation between the nodes subscription and its location. The events are modeled as real numbers. If this number falls in the interval then there is a match, otherwise the subscription and the publication do not match.
- *Topic Trees:* The valid topics are modeled as a hierarchical tree. The leaves of this tree represent the events. The subscriptions are modeled by any node in this tree. This subscription matches all events in the subtree routed in that node.

5.2 Metrics

There is a long list of relevant metrics to choose from:

- *Delivery ratio:* number of received data packets at the subscribers versus the number of data messages sent by the publishers.
- *Distinct-event delivery ratio:* ratio of the number of *distinct* events received to the number originally sent.
- *Multicast performance:* number of data packets transmitted in the network per data packet delivered at the subscribers.
- *Control overhead 1:* number of control and data packets transmitted per data packet delivered.
- *Control overhead 2:* number of control bytes transmitted per data byte delivered (all overhead, including the data headers)
- *Control overhead 3:* bytes of all sent messages on MAC, including just the header of the pub/sub messages divided by the pure payload bytes delivered to the application.
- *Effects of hot-spots:* histogram of sent/received messages per node number.
- *Average dissipated energy:* ratio of total dissipated energy per node to the number of distinct events seen by the subscribers.

References

[1] Antonio Carzaniga, David S. Rosenblum, and Alexander L Wolf. Design and evaluation of a wide-area event notification service. *ACM Transactions on Computer Systems (TOCS)*, 19(3): 332–383, 2001. ISSN 0734-2071.

[2] Yongqiang Huang and Hector Garcia-Molina. Publish/subscribe tree construction in wireless ad-hoc networks. Technical report, Stanford University, Database Group, 2001.

[3] Chalermek Intanagonwiwat, Ramesh Govindan, Deborah Estrin, John Heidemann, and Fabio Silva. Directed diffusion for wireless sensor networking. *IEEE/ACM Transactions on Networking (TON)*, 11(1):2–16, 2003. ISSN 1063-6692.

[4] Virendra Marathe and Ted Herman. Design of interval routing enabled publish/subscribe communication protocol for ad hoc sensor networks. In *Midwest Society for Programming Languages and Systems*, 2002.

[5] Gero Mühl, Ludger Fiege, Felix Gärtner, and Alejandro Buchmann. Evaluating advanced routing algorithms for content-based publish/subscribe systems. In *10th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS 2002)*, 2002.

[6] Lee Sung-Ju, W. Su, J. Hsu, M. Gerla, and R. Bagrodia. A performance comparison study of ad hoc wireless multicast protocols. In *Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM 2000*, volume 2, pages 565–574. IEEE, march 2000.

Mechanisms for Improving Single Packet Delivery Probability in Wireless Sensor Networks

Andreas Willig
Hasso-Plattner-Institute (HPI)
University of Potsdam
awillig@ieee.org

July 5, 2003

1 Introduction

Ultimately, when deploying a wireless sensor network, a user wants to have reliable and credible information. This requires that:

- the desired information is available at all: a user might want to have temperature data from a region where no temperature sensors exist (anymore);
- the sensor readings behind the information are accurate: since sensor nodes are supposed to be rather cheap, sensor readings might be biased
- the sensor readings reach the user (working at a monitor station); and
- the sensor readings reach the user in a timely fashion.

This extended abstract outlines future work regarding the last two bullets, namely, the probability of information delivery and its timing aspects. This work is based on a specific class of sensor networks, namely those which are *data-centric* and use *directed diffusion* [3] or one of its variants (e.g. [4], [5]).

There are different communication scenarios which call for reliable information delivery:

S1 : deliver a single sensor reading (of high importance) reliably to the monitor station or neighboured sensors;

S2 : deliver readings from a group of sensors reliably to the monitor station

S3 : deliver an interest or other data (e.g. files) reliably from the monitor to all sensors of a given type within a region, to a subset of all sensors (reliable multicast), or to all sensors (reliable broadcast)

S4 : deliver data reliably from all sensors to the monitor

In the next future, the first scenario **S1** ("single packet delivery") is considered. In general, the literature on reliability / transport protocols in sensor networks so far is scarce, in [6] the problem of transmitting large data blocks (e.g. program files) to a subset of sensors is attacked. The authors introduce the PSFQ (pump slowly, fetch quickly) reliable transport protocol, which is based on local retransmissions and in-sequence delivery.

2 Goal and Approach

The goal of my upcoming research is to find localized mechanisms to improve the probability of single packet delivery and to evaluate the associated costs of these mechanisms in terms of energy expenditure / network lifetime, as well as their adaptivity to changing network conditions, caused by e.g. topological changes (node death, node mobility), changing error conditions on the wireless links, etc.

There are a lot of basic mechanisms and control knobs which can be used, some of them are:

- MAC-/link layer: increasing transmit power, FEC, ARQ, scheduling for collision reduction / elimination
- network layer / routing: (directed) flooding, provide multiple (totally or partially node-disjoint) paths, packet duplications
- application layer: end-to-end ARQ, error concealment techniques, increasing data rate

If the links have non-negligible packet error rates, end-to-end ARQ is clearly unfeasible, error concealment techniques make only sense for continuous data like multimedia streams (which are not the typical application in sensor networks), and increasing the data rate reduces network lifetime. Therefore, we concentrate on MAC-/link-layer and network layer mechanisms. Existing proposals focus on the network layer:

- in [2] a procedure for constructing *braided multipaths* is proposed to ensure the existence of a path between source and monitor (a similar idea is discussed in).
- The "ReInForM" proposal of [1] is based on the following main ideas:
  - packets are distinguished according to priorities, and high priority packets typically have shorter paths and a higher desired simple packet delivery probability
  - Multiple copies of the same packet are transmitted over multiple (not necessarily edge-disjoint) paths (which are not explicitly constructed but formed probabilistically), each intermediate node again might create multiple copies of the packets. The probability that an intermediate node creates multiple copies is based on reliability information carried in the packet header and on local error rate measurements. The proposed algorithm has the tendency to create multiple copies close to the source, while the rate of copies decays when coming closer to the monitor.

  One of its features is that it does not require intermediate nodes to cache the packets or to keep any further state (e.g. routing information, interest caches).

I want to suggest and investigate another mechanism, which can be roughly described as "copy-on-demand", where the need to copy a packet is signalled reactively from the lower layers after some trials to deliver a packet failed (in the scheme of [1] it is based on history information about previous error rates). Furthermore, the proposed mechanism uses state in intermediate nodes, which, however, is not a problem since in networks based on directed diffusion intermediate nodes keep state anyway, namely their interest caches. The mechanism combines MAC-/link-layer approaches (here: immediate retransmissions) with network-layer approaches (keeping multiple paths and using multiple copies of a packet).

In the following the approach is desribed more detailed. Suppose that some node B wants to transmit a packet to a downstream node C, which is closer to the monitor. Suppose additionally, that B has received the packet from its upstream neighbor A.

- B tries to transmit the packet to C, using a number $k \geq 2$ of trials (immediate MAC layer retransmissions, with acknowledgements). If this fails, the MAC layer of B informs the network layer about this. The number of retransmissions should be small but sufficient to distinguish between transient channel errors and longer bad channel periods.
- If the interest cache of B contains a sufficient number of further possible downstream nodes, B creates a number of copies and sends them to these nodes.

- If there are not enough other downstream nodes, B tries to hand back the packet to A, which then might try alternative routes ("pushback"). To achieve this, B stores for each interest in its interest cache the local address of the last upstream node from which matching data was received.
- The source node might generate a priori a certain number of packets.

I hypothesize that this approach has the following properties, which need to be confirmed by analytical and simulation results:

- This approach creates only the minimum possible number of packets before the first error occurs. If no errors occur at all, the bandwidth and energy expended should be close to minimal.
- Transient channel errors hitting only fractions of a packet do not lead immediately to multiple packets – the small number of retransmissions provides a kind of low-pass filtering for channel fluctuations.
- The nodes are not required to keep a history of channel states; in fact, since wireless channels are spatially diverse, a separate history for each neighbor would be needed, for each of which likely only a few observations are available, which age out. Therefore, such a history is likely inaccurate, and it cannot predict sudden changes from good to bad channel states during the next packet.
- By controlling certain parameters like the number of neighbors to which a copy is transmitted, number of initial copies, etc. the resulting packet delivery probability can be controlled.

In general, i believe that mechanisms for improving reliability should be designed primarily by combining MAC-/link-layer and network layer mechanisms. Application layer mechanisms seem only rarely suited to this task.

3 Research Issues

- Which parameters influence the achievable reliability and energy expenditure of this scheme? Some candidates are: error rates / burstiness / timescales of variation of channel errors; node density and node distribution; average / minimum / maximum node degree;
- How does this scheme compare in reliability / energy expenditure to other schemes, e.g. (directed) flooding, the ReInForM scheme of [1]?
- How can this scheme support different classes of reliability requirements and how can this be integrated with fulfilling timing requirements?
- How can this scheme be used or adapted to implement others of the communication scenarions **S2** - **S4**?
- Can similar schemes be devised without keeping state at intermediate nodes?

References

[1] Budhaditya Deb, Sudeept Bhatnagar, and Badri Nath. Reinform: Reliable information forwarding using multiple paths in sensor networks. In *Proc. 28th Annual IEEE Conference on Local Computer Networks (LCN 2003)*, Bonn, Germany, October 2003. to appear.

[2] Deepak Ganesan, Ramesh Govindan, Scott Shenker, and Deborah Estrin. Highly-Resilient, Energy-Efficient Multipath Routing in Wireless Sensor Networks. *Mobile Computing and Communications Review*, 1(2), 2002.

[3] Chalermek Intanagonwiwat, Ramesh Govindan, Deborah Estrin, John Heidemann, and Fabio Silva. Directed diffusion for wireless sensor networks. *IEEE/ACM Transactions on Networking*, 11(1), February 2003.

[4] Jan M. Rabaey, M. Josie Ammer, Julio L. da Silva, Danny Patel, and Shad Roundy. PicoRadio Supports Ad Hoc Ultra-Low Power Wireless Networking. *IEEE Computer*, 33(7), July 2000.

[5] Rahul C. Shah and Jan M. Rabaey. Energy aware routing for low energy ad hoc sensor networks. In *Proc. IEEE Wireless Communications and Networking Conference (WCNC)*, Orlando, FL, March 2002.

[6] Chieh-Yih Wan, Andrew T. Campbell, and Lakshman Krishnamurthy. Psfq: A reliable transport protocol for wireless sensor networks. In *Proc. First ACM Intl. Workshop on Wireless Sensor Networks and Applications (WSNA'02)*, Atlanta, GA, 2002.