Universität Karlsruhe (TH)
Institut für Telematik

TELEMATICS TECHNICAL REPORTS

2. Fachgespräch der GI/ITG Fachgruppe KuVS "Drahtlose Sensornetze"

26./27. Februar 2004 Universität Karlsruhe (TH)

Thomas Fuhrmann (Editor)
fuhrmann@tm.uka.de

July, 1st 2004

TM-2004-5

ISSN 1613-849X
http://doc.tm.uka.de/tr/

Institute of Telematics, University of Karlsruhe
Zirkel 2, D-76128 Karlsruhe, Germany

2. GI/ITG KuVS Fachgespräch "Drahtlose Sensornetze"

26./27. Februar 2004, Universität Karlsruhe (TH)

Drahtlose Sensornetze haben in letzter Zeit große Beachtung gefunden, einerseits wegen ihres großen Potentials in den verschiedensten Anwendungsbereichen, andererseits aber auch wegen der vielen wissenschaftlich interessanten Fragestellungen, die sich aus den besonderen Anforderungen an diese Netze ergeben. Die Knoten eines Sensornetzes sollten typischerweise klein und sehr kostengünstig herzustellen sein. Häufig sollen sie entweder als Ergänzung in ein Massenprodukt integriert werden oder in großen Mengen in unzugänglichen Gebieten ausgestreut werden. Je nach Anwendungszweck müssen die Sensorknoten daher wartungsfrei und ohne Energiezufuhr für lange Zeit auch unter ungünstigen Umweltbedingungen zuverlässig funktionieren.
Die Reihe dieser Fachgespräche soll aktuelle Ideen und Entwicklungen auf dem Gebiet der drahtlosen Sensornetze durch offene und vielleicht auch kontroverse Diskussionen beleuchten. Sie wird durch ein Leitungsgremium organisiert, dem zur Zeit Dr. Thomas Fuhrman (Universität Karlsruhe), Prof. Kurt Geihs (Technische Universität Berlin), Dr. Holger Karl (Technische Universität Berlin), Prof. Friedemann Mattern (ETH Zürich), sowie Dr. Hartmut Ritter (FU Berlin) angehören.
Wir hoffen, mit der Veranstaltung am 26. und 27. Februar 2004 in Karlsruhe einen Beitrag geleistet zu haben, der dieses interessante und zukunftsweisende Forschungsgebiet wieder ein Stück vorangebracht hat. Unser Dank gilt daher allen Teilnehmern, besonders den Vortragenden, die mit ihren Beiträgen dieses Fachgespräch mitgestaltet haben. Außerdem möchten wir uns bei Curt Cramer und Kendy Kutzner bedanken, die uns vor Ort so tatkräftig bei der Durchführung der Veranstaltung unterstützt haben.

Karlsruhe, im März 2004
Thomas Fuhrmann

Teilnehmer

Erik-Oliver Blaß	Universität Karlsruhe
Jan Blumenthal	Universität Rostock
Alejandro Buchmann	TU Darmstadt
Carsten Buschmann	TU Braunschweig
Georg Carle	Universität Tübingen
Curt Cramer	Universität Karlsruhe
Falko Dressler	Universität Tübingen
Dominique Dudkowski	Universität Stuttgart
Stefan Fischer	TU Braunschweig
Christian Frank	ETH Zürich
Thomas Fuhrmann	Universität Karlsruhe
Matthias Gerlach	Fraunhofer FOKUS, Berlin
Hartwig Grabowski	T-Systems
Stephan Guttowski	Fraunhofer IZM, Berlin
Jörg Hähner	Universität Stuttgart
Till Harbaum	Beecon GmbH, Karlsruhe
Hans-Joachim Hof	Universität Karlsruhe
Klaus Kabitzsch	TU Dresden
Holger Karl	TU Berlin
Jochen Koberstein	Christian-Albrechts-Universität zu Kiel
Andreas Köpke	TU Berlin
Oliver Kraemer	Sony International (Europe) GmbH
Kendy Kutzner	Universität Karlsruhe
Max Larsson	Fraunhofer SIT, Darmstadt
Norbert Luttenberger	Christian-Albrechts-Universität zu Kiel
Pedro-Jose Marron	Universität Stuttgart
Friedemann Mattern	ETH Zürich
Daniel Minder	Universität Stuttgart
Gero Mühl	TU Berlin
Gerhard Münz	Universität Tübingen
Mario Neugebauer	TU Dresden
David-Dmitry Polityko	Fraunhofer IZM, Berlin
Frank Reichenbach	Universität Rostock
Matthias Ringwald	ETH Zürich
Hartmut Ritter	FU Berlin
Kay Roemer	ETH Zürich
Katja Schwieger	TU Dresden
Matthias Transier	Universität Mannheim
Volker Turau	TU Hamburg-Harburg
Christoph Weyer	TU Hamburg-Harburg
Martina Zitterbart	Universität Karlsruhe

Inhaltsverzeichnis

Parameteroptimierung grobkörniger Positionierungsalgorithmen in Sensor-Netzwerken

Jan Blumenthal, Frank Reichenbach, Dirk Timmermann
Institut für Angewandte Mikroelektronik und Datentechnik
Universität Rostock
Richard-Wagner-Str. 31, 18119 Rostock, Deutschland
{jan.blumenthal, frank.reichenbach, dirk.timmermann}@etechnik.uni-rostock.de

Abstrakt

Dieser Artikel beschreibt unsere Forschungsaktivitäten auf dem Gebiet der grobkörnigen Positionsbestimmung in Sensor-Netzwerken. Unsere aktuellen Untersuchungen konzentrieren sich auf die Optimierung der einstellbaren Übertragungsreichweite von Basisstationen beim Senden von Positionsdaten und die Optimierung der Positionsberechnung auf den einfachen Knoten mit dem Ziel der Minimierung des Energieaufwands.
In diesem Artikel präsentieren wir erste Ergebnisse, wie für den grobkörnigen Algorithmus „Coarse Grained with Center Determination" eine optimale Übertragungsreichweite eingestellt werden kann.

1 Einleitung

Ein Sensor-Netzwerk besteht aus hunderten oder tausenden winziger Sensorknoten, die auf einem Gebiet zufällig verteilt werden. Die Aufgabe besteht in der Aufnahme von Messwerten und der effizienten Weiterleitung an die Datensenken des Systems. Einen sehr guten und einführenden Überblick in die Thematik der Sensor-Netzwerke gibt I.F. Akyildiz in [1].
Ein Sensorknoten besteht aus einer Batterie, einem Mikroprozessor, einer Übertragungseinheit und einem Sensor. Aufgrund der Größe eines Knotens von einigen Kubikmillimetern sind die Dimensionen für die Konstruktion von Übertragungseinheit und Batterie besonders kritisch. Die knappste Ressource innerhalb eines Netzwerkes ist folglich die zur Verfügung stehende Energie. Es ist demnach unerlässlich, neben stromsparenden Bauelementen auch energieeffiziente Algorithmen einzusetzen.
Die stochastische Verteilung von Sensorknoten verhindert eine sofortige Zuordnung des Messdatums zum Ort der Messung. Aus diesem Grund ist eine Positionsbestimmung der Knoten notwendig, die zusätzliche Energie für Datenübertragungen und Berechnungen auf den einfachen Knoten benötigt.

2 Positionsbestimmung

Die Positionsbestimmung in Sensor-Netzwerken wird in zwei große Teilbereiche unterschieden: die grob- und die feinkörnige Positionsbestimmung.
Die feinkörnige Positionsbestimmung zeichnet sich durch sehr exakte Ergebnisse mit einem sehr geringen Fehler (<5%) aus. Ein sehr bekannter Algorithmus ist die Multilateration, die von A. Savvides vorgestellt wurde [2]. Der wesentliche Nachteil der feinkörnigen Positionierung besteht in dem hohen Aufwand für Berechnung und Kommunikation. Zudem werden zumeist hohe Anforderungen an die Qualität der Messwerte, z.B. der Feldstärkemessung, gestellt, die sich in der Praxis als unrealistisch erweisen.
Die grobkörnige Positionsbestimmung unterscheidet sich von feinkörnigen Ermittlungsmethoden durch einfachere Modelle, geringeren Kommunikationsaufwand und schnellere Berechnung. Der Nachteil besteht in den höheren Lokalisationsfehlern, die zwischen 5-20% liegen, aber für die meisten Anwendungen in Sensor-Netzwerken ausreichend sind.
Basierend auf Infrarot-Übertragungstechnik wurde bereits 1989 das „Active Badge System" für Innenräume erfolgreich implementiert [3]. Das hier betrachtete Verfahren

basiert auf dem von Nirupama Bulusu vorgestellten „Coarse Grained with Center Determination“ Algorithmus [4].
Die Basis der Positionsbestimmung bildet ein idealisiertes Radiomodell, um die Grenzen der Kommunikationsreichweite einzelner Knoten zu bestimmen. Sendeaktivität findet lediglich durch Basisstationen, nachfolgend Beacons genannt, statt. Perfekte kugelförmige Radiowellenausbreitung mit dem Radius R und identische Sendebereiche für alle Transceiver ergänzen das idealisierte Modell.
Als Vereinfachung betrachten wir ein zweidimensionales, quadratisches Feld mit b Beacons. Die Beacons $B_1 ... B_b$ sind in einem Gitternetz (Infrastrukturfall) mit einem homogenen Abstand d zueinander angeordnet. Sie befinden sich an den bekannten Stellen $(x_1 ... x_b),(y_1 ... y_b)$ und senden zeitlich versetzt in periodischen Abständen ein Signal in Form eines Paketes aus, das von den in Reichweite befindlichen einfachen Knoten mit unbekannter Position (Unbekannten) empfangen wird. Jeder Unbekannte überwacht das Übertragungsmedium für eine festgelegte Zeit t und registriert alle empfangenen Pakete. Anschließend bestimmt er die Anzahl n der sich in Reichweite befindlichen Beacons.

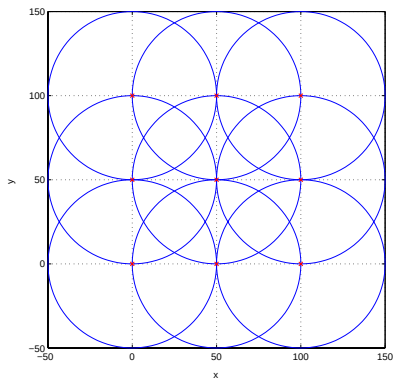

Abb. 1: Beaconverteilung mit d = 50 und R = 50

Mit der Annahme eines idealen kreisförmigen Sendebereiches der Beacons ist eine Aufteilung des Feldes in Überlappungsbereiche möglich. In Abhängigkeit der Verteilung der Beacons entstehen Regionen mit $0 ... b$ Überlappungen (Abbildung 1). Jeder Knoten innerhalb eines Überlappungsbereiches kann durch einfache Mittelpunktbestimmung der empfangenen Beaconpositionen seinen Schätzwert der eigenen Position bestimmen. Mithilfe der Gleichung 1 ist dieser Schätzwert als Mittelpunkt innerhalb der eingeschlossenen Fläche ermittelbar.

$$x_{i_{est}}, y_{i_{est}} = \left(\frac{\sum_{k=1}^{n} x_{B_k}}{n}, \frac{\sum_{k=1}^{n} y_{B_k}}{n} \right) \quad (1)$$

$x_{i_{est}}, y_{i_{est}}$ = geschätzte Koordinaten

Der Schätzwert ist für alle Knoten innerhalb einer Überlappungsregion konstant, im Gegensatz zum Positionierungsfehler. Er ist definiert als Abstand zwischen der geschätzten und der exakten Position des Unbekannten. Diese Beziehung drückt Gleichung 2 aus.

$$f_i(x,y) = \sqrt{\left(x_{i_{est}} - x_{i_a}\right)^2 + \left(y_{i_{est}} - y_{i_a}\right)^2} \quad (2)$$

$f_i(x,y)$ = Positionierungsfehler; x_{i_a}, y_{i_a} = Koordinaten

Die Genauigkeit der Positionsbestimmung wird nach N. Bulusu [4] mit zunehmendem Verhältnis von r zu d besser. Unsere Simulationen zeigen jedoch ein anderes Verhalten. Zuerst steigt die Genauigkeit mit zunehmender Beaconanzahl, aber anschließend sinkt sie wieder. Es scheint ein Optimum in der Anzahl der Beacons zu geben, das zudem abhängig von der Übertragungsreichweite der Beacons ist.
Unsere Untersuchungen konzentrieren sich somit auf die Möglichkeit, durch Optimierung der Übertragungsreichweite Energie einzusparen. Das ist besonders vorteilhaft, da der Energieverbrauch beim Senden von Nachrichten quadratisch mit der Entfernung r wächst, wie aus der vereinfachten Gleichung 3 zur Energiebilanz einer Funkübertragung zu erkennen ist.

$$E = E_{Init} + nE_{Bit}\left(\frac{4 \cdot \pi \cdot r}{\lambda}\right)^2 \quad (3)$$

n = Anzahl der zu übertragenden Bits
E_{Init} = Basisbedarf des Transmitters
E_{Bit} = Energie zur Übertragung eines Bits
λ = Wellenlänge
r = Distanz

3 Aktuelle Untersuchungen

3.1 Gitternetzausrichtung der Beacons

Die durchgeführte Simulation umfasste ein quadratisches Gitterfeld 100·100. Wichtig war uns, die Abhängigkeit der Übertragungsreichweite R von der Beaconanzahl und dem resultierendem Fehler zu untersuchen

(Abbildung 2). Es ist ersichtlich, dass mit ansteigender Reichweite ein anfangs sehr hoher Fehlerwert sprunghaft auf sein Minimum absinkt. Danach steigt der Fehler fast linear mit der Reichweite an. Ebenso ist zu erkennen, dass eine höhere Beaconanzahl bei konstanter Übertragungsreichweite keinesfalls immer zu einem geringeren Fehler führt.
Im Idealfall wird eine hohe Genauigkeit mit wenig Beacons und einer möglichst geringen Übertragungsreichweite gefordert. Nach Abbildung 2 nimmt die optimale Reichweite mit steigender Anzahl von Beacons stetig ab und reduziert den Fehler bei 10·10 Beacons bis auf 1,89.
Es entsteht somit ein Optimierungsproblem zwischen Beaconanzahl, dem Fehler und der Übertragungsreichweite. Abbildung 2 gibt weiterhin Einsicht in die Fehlerentwicklung bei Erhöhung der Reichweite im Allgemeinen. Eine Überdimensionierung des Übertragungsbereiches der Beacons ist nicht sinnvoll und führt zu einer Erhöhung des Fehlers und des Energieverbrauchs. Der auffällige Funktionsverlauf bei vier verwendeten Beacons beweist ein unausgeglichenes Verhältnis zwischen Reichweite, Beacons und der Feldgröße. Eine zu geringe Anzahl von Beacons wird daher nicht empfohlen.

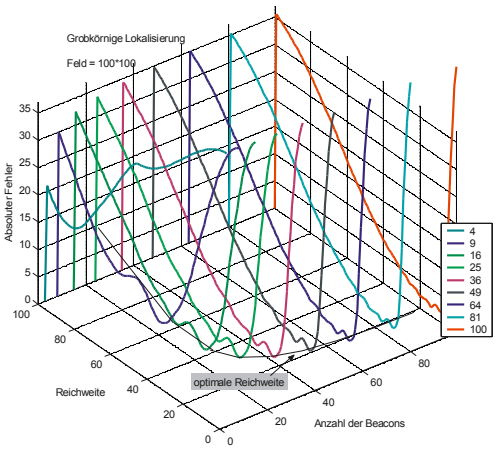

Abbildung 2: Abhängigkeit des absoluten Fehlers von der Übertragungsreichweite und der Anzahl der Beacons (Gitternetz-Anordnung)

3.2 Stochastische Beacon-Verteilung

Folgend wird die Eignung des Simulationsverfahrens auf Basis stochastisch verteilte Beacons untersucht. Es ist anzunehmen, dass die Resultate nicht sehr exakt sein werden. Zur Verifikation diente eine Simulation mit 36 zufällig gleichverteilten Beacons.

Bei jeweils konstanter Anzahl Beacons wurden drei verschiedene Verteilungen in einem Feld in Abhängigkeit der Übertragungsreichweite aufgenommen. Das Resultat ist in Abbildung 3 dargestellt. Obwohl die Beacons bei jeder Kurve unterschiedliche Positionen besitzen, entstehen ähnliche Verläufe. Bei *R=24* bildet sich eine optimale Übertragungsreichweite aus. Der Kurvenverlauf des absoluten Fehlers gleicht dem im Infrastrukturfall. Die Kurve fällt schnell auf ein Minimum ab und steigt anschließend fast linear an.

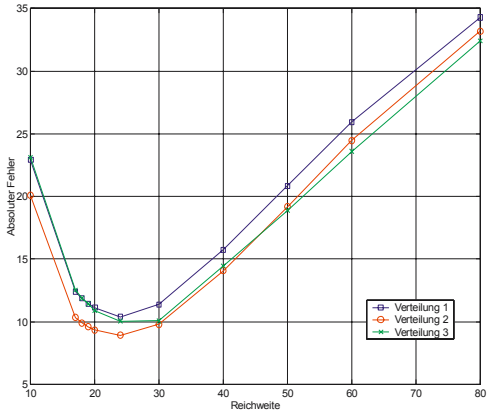

Abbildung 3: Abhängigkeit des absoluten Fehlers von der Übertragungsreichweite bei verschiedenen stochastischen Verteilungen

Die stochastische Verteilung der Beacons führt zu einer Erhöhung der optimalen Übertragungsreichweite sowie einer Verschlechterung der Genauigkeit. Der Fehler erreicht ein Minimum zwischen 7,9 und 9,4 gegenüber 3,3 bei Gitteranordnung der Beacons. Durch den zufälligen Einfluss kann demnach mit einer Verdreifachung des Fehlers gerechnet werden.

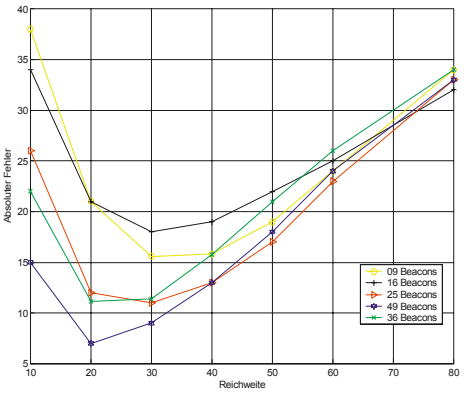

Abbildung 4: Abhängigkeit des absoluten Fehlers von der Übertragungsreichweite und der Anzahl der Beacons (stochastische Verteilung)

Weiterhin wurde eine Simulationsreihe angefertigt mit einer variablen Anzahl von Beacons (Abbildung 4). Dabei verbesserte sich der Fehler mit Erhöhung der Beaconanzahl. Dieses Verhalten gleicht ebenfalls den Erfahrungen aus dem Infrastrukturfall.

4 Zusammenfassung

Wir präsentierten in diesem Artikel erste Ergebnisse beim Einstellen der optimalen Übertragungsreichweite von Beacons, um den Energieaufwand während der Positionsbestimmung der Knoten eines Sensor-Netzwerkes zu verringern.
Anhand der Simulationsergebnisse kann für jede Kombination von Beacons zu einfachen Knoten in Abhängigkeit des gewünschten Positionierungsfehlers die optimale Übertragungsreichweite der Beacons abgelesen werden.
Aktuelle Forschungsaktivitäten betreffen die Ermittlung der optimalen Übertragungsreichweite für die stochastischen Verteilung von Knoten. Weiterhin konzentrieren wir uns auf eine analytische Lösung des Optimierungsproblems zwischen Beaconanzahl, Positionierungsfehler und Reichweite.

Literatur

[1] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E.Cayirci: „*A Survey on Sensor Networks*“, IEEE Communications Magazine, pp. 102-114, August 2002

[2] Andreas Savvides: “*Dynamic fine grained localization in ad-hoc networks of sensors*”, ACM SIGMobile 07/2001, Rom, 2001

[3] Want Roy: “*The active badge location system*”, ACM Transactions on Information Systems, 10(1):91-102, January 1992

[4] Nirupama Bulusu, John Heidemann, Deborah Estrin: „*GPS-less low cost outdoor localization for very small devices*”, 2000

Eigenschaften der Funkschnittstelle in Sensornetzen und Auswirkungen für Anwendungen

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

In Sensornetzen gibt es einen starken Zusammenhang zwischen den Eigenschaften verschiedener Schichten des Protokollstapels und den Applikationen. Dies liegt unter anderem an der physikalischen Einbettung der Sensorknoten in ihre Umwelt und drückt sich beispielhaft in der Wechselwirkung von Eigenschaften der Funkschnittstelle und der Funktionsweise insbesondere von Applikationen aus, für die Lokationsbewusstsein von essenzieller Bedeutung ist.

Related Work. Es hat in den vergangenen Jahren zahlreiche Publikationen hinsichtlich Verfahren zur Errichtung von Lokationsbewusstsein gegeben. Verschiedene Verfahren setzen dabei auf Entfernungsschätzungen zu Nachbarn auf Basis von Signalstärkemessungen der von benachbarten Geräten empfangenen Funksignale (beispielhaft [BM02], [PDW01]). Im Gegensatz zur Verwendung spezieller Hardware zur Bestimmung von Position oder Entfernung hat dies den Vorteil, dass man Kosten, Baugröße und Energie sparen kann. Es stellt sich jedoch die Frage nach der Genauigkeit der erzielbaren Ergebnisse. In der Forschungsgemeinde gibt es durchaus ein Bewusstsein für die damit verbundenen Schwierigkeiten im Hinblick auf Lokationsverfahren, wie die Justierungsbemühungen beispielsweise in [SRL02] zeigen. Ein weiterer Ansatz zur Nutzung der Funkschnittstelle ist die Variation der Sendeleistung zur Einstellung des Kommunikationsradius (z.B. [DPG01]). Obwohl in [GKW+03] der Kommunikationsbereich von Sensorknoten untersucht wurde, sind Arbeiten in diesem Bereich weiterhin die Ausnahme. So werden in [B99] zwar Eigenschaften von WLAN Interfaces untersucht, lediglich [W02] berichtet von Experimenten mit in Sensorknoten üblichen Transceivern.

Grundsätzlicher Aufbau. In diesem Beitrag werden zwei Versuchsreihen sowie deren Ergebnisse vorgestellt, die sich mit dem Zusammenhang zwischen gemessener Empfangssignalstärke und Entfernung sowie Sendeleistung und Reichweite beschäftigen. Zum Einsatz kamen die von der FU Berlin entwickelten Embedded Sensor Boards ESB 430/1 [ESB03] (siehe Abb. 1). Sie bieten sowohl die Möglichkeit, die Sendeleistung prozentual zu variieren, als auch eine Schnittstelle zum Auslesen der Empfangssignalstärke. Dazu ist ein analoger Ausgang der Funkschnittstelle auf einen AD-Wandler geführt. Dieser Mechanismus wird auch zur Unterscheidung zwischen Rauschen auf dem Empfänger und tatsächlichen Paketen verwendet, so dass ein Schwellwert zu definieren ist, unterhalb dessen Signalstärkenmessungen nicht möglich sind. Es werden Untersuchungen in Korridoren und Versuche mit Freifeldcharakteristik unterschieden. Letztere wurden im Sommer begonnen. Da aufgrund der winterlichen Temperaturen in letzter Zeit Außenversuche nicht möglich waren, sind wir auf eine Turnhalle ausgewichen. Bei Vergleich der Ergebnisse von wirklichem Freifeld und Turnhalle hat sich eine sehr starke Ähnlichkeit herausgestellt, so dass diese als vergleichbar angesehen werden können.


Abb.1: ESB 430/1

Empfangssignalstärkenmessungen. In einer ersten Versuchsreihe wurde der Zusammenhang zwischen dem Abstand zwischen Sender und Empfänger einer Nachricht und der Empfangssignalstärke untersucht. Dazu wurden beide Geräte in einem Abstand zwischen ein und neun Metern voneinander in einer Höhe von einem Meter über dem Boden positioniert. Der Sender verschickt in einem Abstand von 400 Millisekunden Pakete, der Empfänger protokolliert zu jedem empfangenen Paket die gemessene Signalstärke sowie eventuelle Bitfehler.

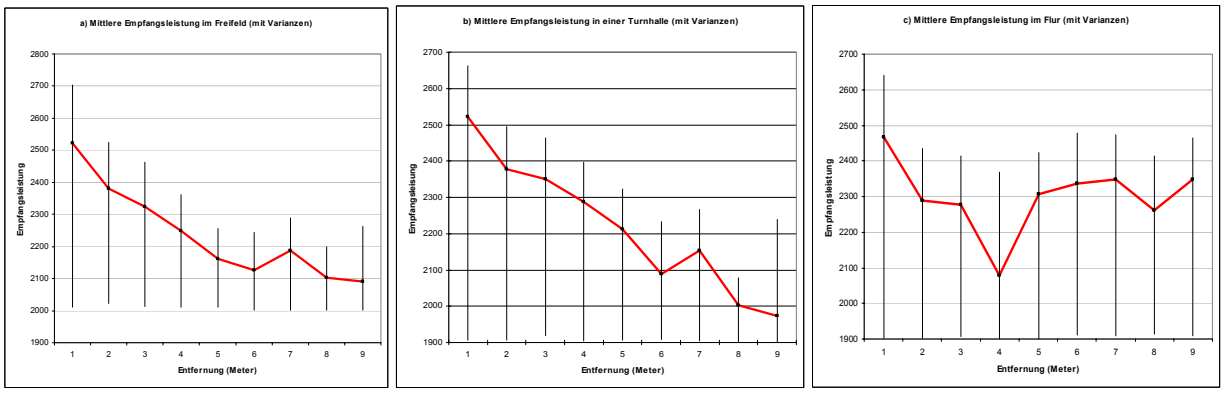

Abb. 2: Zusammenhang zwischen Empfangssignalstärke und Entfernung

Der Versuch wurde sowohl im Freifeld in einem Braunschweiger Park, in einer Turnhalle als auch in der Mitte eines Flurs im Institut für Betriebssysteme und Rechnerverbund der TU Braunschweig durchgeführt.

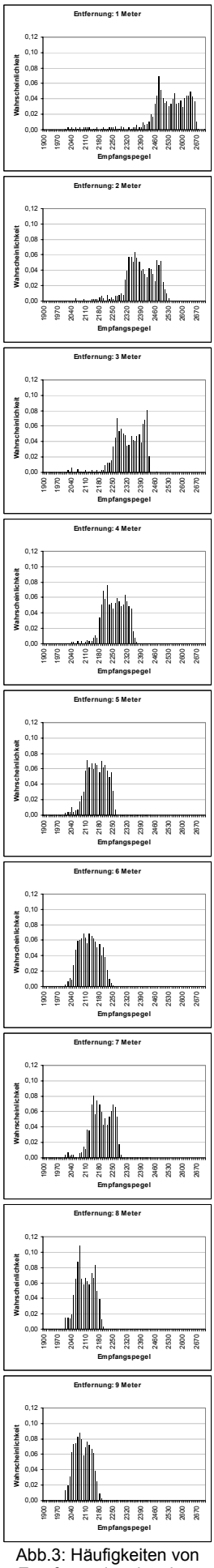

Abb.3: Häufigkeiten von Empfangssignalstärken

Es stellte sich erwartungsgemäß heraus, dass die gemessene Empfangssignalstärke grundsätzlich mit zunehmender Entfernung abnimmt. Abbildung 2 zeigt den Zusammenhang zwischen Entfernung und den mittleren Messwerten (sowie deren Minimal- und Maximalwerte) in Freifeld (a), Turnhalle (b) und Flur (c). Es wird sofort klar, dass in letzterer Umgebung sinnvolle Entfernungsschätzungen aus den Messwerten nicht möglich sind, da hier die Empfangssignalstärke weniger von der Entfernung als von den Gegebenheiten der Umgebung abhängt. Eine Übersicht über Gründe dafür gibt [MGE+02]. Auf Details der Umgebung wird im folgenden Abschnitt noch genauer eingegangen.

Die Diagramme (a) und (b) zeigen eine große Ähnlichkeit. Dennoch ist auch in diesen Fällen eine sinnvolle Entfernungsschätzung kaum möglich. Der Grund dafür liegt in der hohen Varianz der Messwerte und wird in Abbildung 3 sichtbar. Sie zeigt Häufigkeitsverteilungen der Empfangssignalstärken bei verschiedenen Entfernungen im Freifeld; es handelt sich somit lediglich um eine andere Darstellung der Daten aus Abbildung 2(a). Es ist klar erkennbar, dass die gemessenen Signalstärken auch bei gleich bleibender Entfernung erheblich schwanken. Somit überlappen sich die Bereiche, in denen bei den verschiedenen Entfernungen häufig Messwerte auftreten, erheblich. Selbst wenn man eine größere Anzahl von Messwerten heranzöge, ließe sich bestenfalls eine Entfernung kleiner einem Meter von den anderen Fällen unterscheiden. An den Messwerten für 7 Meter ist erkennbar, dass die Signalstärken mit zunehmender Entfernung durchaus auch wieder ansteigen können. Warum dies in beiden Fällen bei 7 Metern geschieht, ist zurzeit noch ungeklärt.

Aufgefallen ist ebenfalls, dass bei den Versuchen im Flur die Rate der mit Bitfehlern behafteten Pakete mit 2% deutlich geringer ausgefallen ist als in der Turnhalle (22%). Da die Häufigkeiten der Signalstärken innerhalb von fehlerhaften und fehlerfreien Paketen allerdings ähnlich verteilt sind, sind Pakete mit Bitfehlern zur Entfernungsschätzung genauso (un-)geeignet wie solche ohne Fehler. Aufgrund eines Softwarefehlers stehen für die Freifeldmessungen keine zuverlässigen Informationen über die Bitfehlerraten zur Verfügung.

Sendeleistungsmessungen. Nachdem sich die Messung von Empfangssignalstärken für die Entfernungsschätzung als wenig geeignet herausgestellt hatte, untersuchten wir den Zusammenhang zwischen der Entfernung zwischen Sender und Empfänger und der notwendigen Sendeleistung, um diese zu überbrücken. Die Geräte wurden erneut in einer Höhe von einem Meter in einem Abstand zwischen wenigen Zentimetern und 17 Metern positioniert, und der Sender beginnt mit zunehmender Leistung zu senden. Empfängt der Empfänger ein Paket, sendet er es mit voller Leistung zurück. Auf diese Weise kann der Sender protokollieren, ab welcher Sendeleistung er den Empfänger erreicht. Dieser Ablauf wurde hundertfach wiederholt und aus den Messwerten Mittelwert, Minimum und Maximum abgeleitet. Abbildung 4 zeigt die Ergebnisse dieses Versuches in einem Flur des Instituts für Betriebssysteme und Rechnerverbund in einem Abstand von etwa 20cm von der Wand. Um die Ergebnisse ins Verhältnis zur Umgebung setzen zu können, ist die betreffende Wand ebenfalls abgebildet.

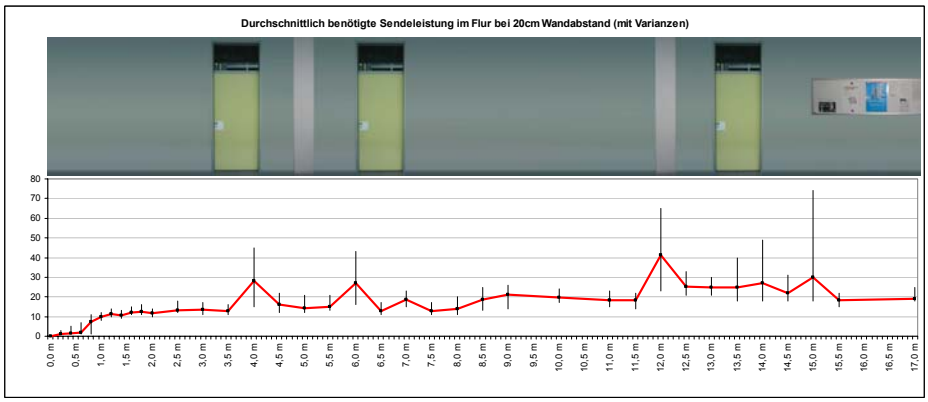

Abb. 4. Mittlere benötigte Sendeleistung

Erwartungsgemäß steigt die erforderliche Sendeleistung mit zunehmender Entfernung. Es ist auffällig, dass die benötigte Sendeleistung maßgeblich von der Stelle abhängig ist, an der sich der Empfänger befindet. Wurde er in der Nähe von Türrahmen (4m bzw. 6m) oder Stahlbetonsäulen (12m) positioniert, war wie erwartet eine erheblich höhere Sendeleistung erforderlich, um ihn zu erreichen. Bemerkenswert ist jedoch, dass der Einfluss dieser Faktoren sehr unterschiedlich sein kann. So wirkt sich die Säule bei einer Entfernung von 5 Metern ebenso wenig aus wie die Türrahmen bei etwa 13 bzw. 14 Metern. Warum dies so ist, ist zurzeit noch unklar. Auffällig war jedoch, dass es für eine Reproduktion der Messergebnisse erforderlich war, die Positionierung der Geräte sehr exakt zu wiederholen, da bereits Positionsabweichungen von zwei Zentimetern zu stark veränderten Ergebnissen führten. Wiederholt man die Versuchsreihe in der Mitte des Flurs anstatt dicht an der Wand, sind die Ergebnisse ähnlich, allerdings zeigen sich die einzelnen Peaks in der Messkurve etwas weniger ausgeprägt.

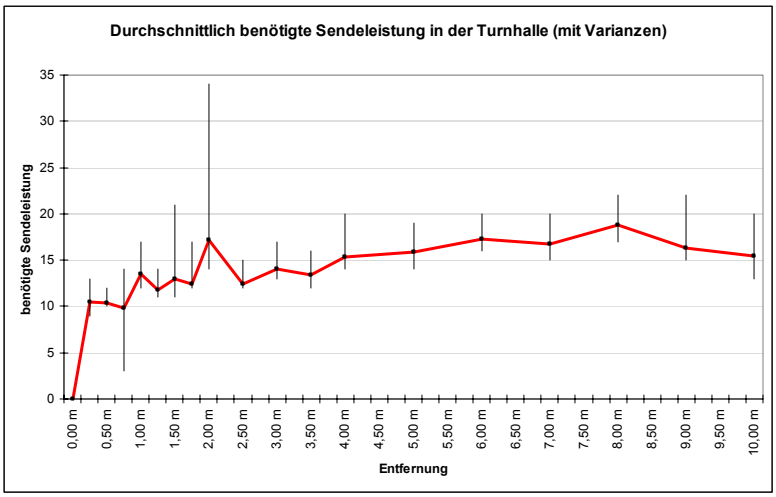

Abb. 5: Mittlere Sendeleistung in einer Turnhalle

Bei beiden Versuchsreihen lässt sich sagen, dass wiederum der Nahbereich von größeren Entfernungen zu unterscheiden ist, ansonsten aber aus den Messwerten kaum Aussagen über Entfernungen möglich sind, da die benötigten Sendeleistungen recht stark schwanken.

Abbildung 5 zeigt die Ergebnisse einer Wiederholung des Experimentes in einer Turnhalle. Es ist erkennbar, dass der Kurvenverlauf wesentlich gleichmäßiger ist. Dennoch zeigen sich bei 75 cm und bei 2 Metern Ausschläge nach unten bzw. oben insbesondere der Minima/Maxima, ohne dass dafür in der Umgebung Ursachen erkennbar waren. Insgesamt liegen die benötigten Sendeleistungen etwas höher als im Flur, was wohl durch fehlende Reflektionen zu erklären ist. Ansonsten ähneln die Ergebnisse denen aus den Messungen in den Fluren. Messungen im Freifeld wurden nicht durchgeführt.

Zusammenfassung. Es wurden Ergebnisse von Versuchsreihen vorgestellt, die sich mit dem Zusammenhang zwischen gemessener Empfangssignalstärke und Entfernung sowie Sendeleistung und Reichweite beschäftigen. Dabei wurden erhebliche Schwankungen der Messwerte festgestellt. Entfernungsschätzungen auf Basis solcher Messungen sind daher derartigen Ungenauigkeiten unterworfen, dass sinnvolle Aussagen unmöglich erscheinen. Lediglich der Nahbereich lässt sich von größeren Entfernungen unterscheiden.

[B99] J. Beutel. Geolocation in a Picoradio Environment. Master Thesis, ETH Zurich and UC Berkeley, December, 1999

[BM02] P. Bergamo, G. Mazzini: Localization in Sensor Networks with Fading and Mobility, PIMRC 2002, Lisboa, Portugal, September 15-18 2002

[DPG01] Lance Doherty, Kristofer S. J. Pister, and Laurent El Ghaoui. Convex position estimation in wireless sensor networks. In Proceedings of IEEE Infocom 2001, volume 3, pages 1655-1663. IEEE, IEEE Computer Society Press, April 2001.

[ESB03] Webseite der Embedded Sensor Boards der FU Berlin: http://www.inf.fu-berlin.de/inst/ag-tech/esb/english/index.htm

[GKW+03] Deepak Ganesan, Bhaskar Krishnamachari, Alec Woo, David Culler, Deborah Estrin and Stephen Wicker: Complex Behavior at Scale: An Experimental Study of Low-Power Wireless Sensor Networks, Submitted for review to INFOCOM 2003, July 2002. UCLA Computer Science Technical Report UCLA/CSD-TR 02-0013

[MGE+02] William Merrill, Lewis Girod, Jeremy Elson, Kathy Sohrabi, Fredric Newberg, and William Kaiser: Autonomous Position Location in Distributed, Embedded, Wireless Systems, IEEE CAS Workshop on Wireless Communications and Networking, Pasadena, September 2002

[PDW01] Neal Patwari, Robert J. O´Dea, Yanwei Wang: Relative Location in Wireless Networks, IEEE Vehicular Technology Conference, Rhodes, Greece, May 2001

[SRL02] Chris Savarese, Jan Rabaey, Koen Langendoen: Robust Positioning Algorithms for Distributed Ad-Hoc Wireless Sensor Networks, Proceedings of the General Track: 2002 USENIX Annual Technical Conference, Pages: 317 – 327, 2002

[W02] Kamin Whitehouse: The Design of Calamari: an Ad-hoc Localization System for Sensor Networks, Master's Thesis, University of California at Berkeley, 2002

Multi-Hop Transmission: Benefits and Deficits

Katja Schwieger and Gerhard Fettweis
Vodafone Stiftungslehrstuhl Mobile Nachrichtensysteme
Technische Universität Dresden
Email: {schwieg, fettweis}@ifn.et.tu-dresden.de

1 Why (not) multi-hop?

Sensor networks are supposed to operate with very little energy, are build of tens to thousands nodes and may be spread over large distances. In order to interconnect nodes in large areas while spending little energy, a self-evident approach is to use sensors as relays, thus saving transmit power while improving connectivity. In other words, nodes are not necessarily directly connected to a base station, but may use multiple hops to transmit their data. Extensive research has been done in that area for high rate, ad-hoc networks. Nevertheless, the main focus in these works is on network capacity [2] and bandwidth efficiency. As sensor networks are supposed to work at low rate with low duty cycles, these metrics are not nearly as important as energy efficiency. Even if energy is an issue, often only transmit power is considered, neglecting the enormous influence of receive energy and fixed costs. A first approach taking into account those can be found in [1]. Min [6] did a more sophisticated analysis.

So why want we use multi-hop schemes at all? Looking at the exponential path loss model it is immediately obvious, that with more hops n we can save lots of energy:

$$P_{Loss} \sim \left(\frac{d}{n}\right)^{\alpha}, \qquad (1)$$

where α denotes the path loss index and P_{Loss} the overall power loss at a distance d. Thus, we can save $10\alpha \lg(n)$dB. In networks, where interference is an issue, the reduced transmit power yields also reduced interference, improving the energy balance even more. But the main advantage of allowing multi-hop is its capability to avoid hidden terminals. Nodes, which can not reach the base station directly are given the possibility to access it via other nodes, making a much larger network feasible.

But this is not the whole truth. Even though the source node may save energy, relaying nodes have to spend transmit energy as well. Moreover, in order to receive the packets properly, the relays have to be in receive mode for some time. Of course, several MAC-schemes were developed to reduce idle listening and overhearing [8], [4]. Nevertheless, even in fully synchronized networks, where all nodes know their neighbors and their duty cycles, the receive energy and transmit energy have to be included in the calculation. Additional communication overhead has to be considered as well. Even worse, nodes close to the base station would have to handle more traffic, leading to an energy-unbalanced system, causing a decreased network lifetime. So the overall energy consumption for a data transmission using multiple hops may be worse than for the single-hop case.

But what are the important parameters, which determine the usefulness of multi-hop? As it is obvious from (1) the distance between source and destination (as well as between the hops) and path loss index are crucial factors. Furthermore, the relation between the cost of receive and transmit power will influence the expediency of multi-hop. Data fusion can have considerable benefits by compressing data from many sources, yielding condensed data. Very much important is the share of the power amplifier to the fixed costs while transmitting. If the power amplifier needs only a small percentage of the energy, then it would be better to spend more power for a single-hop connections, avoiding costs for other nodes. All these parameters have to be considered under different channel conditions; moreover other channel assumptions may have to be made for single- and multi-hop schemes.

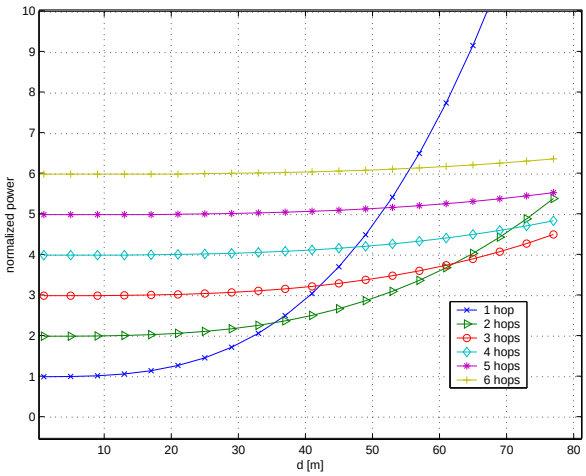

Figure 1: Power consumption for multi-hop, $\alpha = 3$ if $E_{RX} = 0$

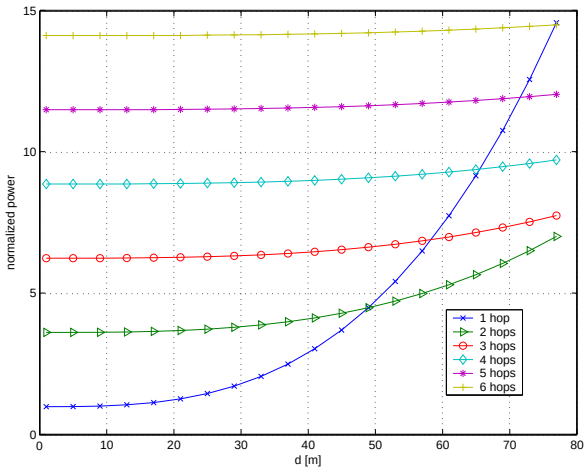

Figure 2: Power consumption for multi-hop, $\alpha = 3$ if $E_{RX} \neq 0$)

2 Possible Approaches

2.1 Theoretical Approach

Let us start with a very simple model for the overall energy consumption when transmitting a single packet as already proposed in [1]. For conventional relaying, illustrated in Fig. 3a), the packet is received by a relay, which will simply retransmit it. For a first estimation, hops with equal distance are assumed. If the final receiver is a power-independent base station (BS), we find

$$E(n, d, \alpha) = n\, E_{TX} + (n-1)\, E_{RX} \qquad (2)$$
$$= n(E_0 \left(\frac{d}{n d_0}\right)^{\alpha} + E_{fix}) + (n-1)E_{RX}.$$

Herein, the overall energy E is a function of the number of hops n, the path loss index α, ranging from 1.5 to 6 according to [3], the distance d and also depends on the energy consumption in receive mode and a fixed share in transmit mode, E_{RX} and E_{fix}, respectively. E_0 is the radiated energy at 1m distance to the transmitter.

Looking at an indoor scenario, a rough estimate about the efficiency of multi-hop protocols can be obtained. We apply a link budget with parameters provided in the table in 4. Furthermore, the power consumption characteristics of Chipcon CC1000 are oriented on. This model assumes fixed energy costs for low transmit power ($\leq$-20 dBm) and a linear increase in power consumption for high radiated powers at a linear scale (i.e. in W). In Fig. 1 and 2 we look at the power consumption, normalized to the consumption for single-hop at 1m, as function of distance for several hops, which are assumed to be equally spaced. The former sets E_{RX} in (2) to 0, i.e. it assumes that no energy is needed for receiving. This gives us a lower bound for the distance, where multi-hop is superior to single-hop. For Fig. 2 a receive power of 42mA according to Chipcons specifications is presumed. Clearly, multi-hop is only useful for distances larger than 35m and 50m, respectively. For indoor scenarios those large gaps are rather rare.

More sophisticated approaches can be adapted from well-established high data rate ad-hoc networks. The idea is to consider more elaborated schemes like cooperative relaying [5]. As shown in Fig. 3b) the data is sent to a relay. At the same time the base station can also receive that signal. Depending on the quality of the signal at the relay, data might be forwarded to the BS, which will combine both signals. This method exploits the advantages of reduced path losses and spatial diversity. Several modifications are known, which differ in the decision *when* to forward and *how* to forward (amplify&forward or decode&forward). Drawbacks are possible error propagations, required channel state information (which might be costly to gather)

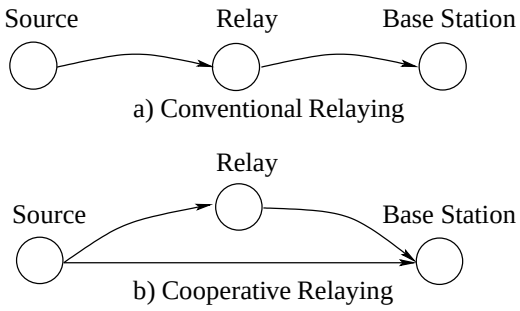

Figure 3: Conventional vs. Cooperative Relaying

and how to divide the power between source and relay node. Instead of using performance measures like outage probability, capacity, spectral efficiency etc. our new measure will be the energy consumption of all relaying nodes and the probability of a successful decoding of the data at the base station. The calculation of possible transmit energy savings has to be counterbalanced against the costs of the relay for receiving, computation and forwarding.

2.2 Practical Approach

Real networks are much more complex. The above described theoretical approach can provide a bound, what can be achieved with multi-hop. It assumes perfect knowledge about routing path, the channel, disregards overhearing and idle listening etc. For practical analysis these and many more parameters have to be regarded. These include retransmission schemes, the transmission of acknowledgments, employment of error correcting codes, physical transmission parameters. Due to those it can be expected that the brute reality may look much worse than the results from theory may suggest. To a certain extend the analysis model proposed in [7] can be used to estimate multi-hop expenses in real networks. Furthermore, simulations with real network protocols would gain a deeper insight.

3 Quo vadis?

Obviously, first rough calculations do not clearly vote for multi-hop schemes. Nevertheless, it remains unclear how the various parameters influence the overall energy consumption of the nodes and the cost for a single transmission. However, in order to provide good coverage for large scaled networks multi-hop may be just inevitable. Thus, we have to be interested in: How much does it cost? So putting up this question we now have a new focus of our research.

4 Appendix

Transmission Rate	250 kbps
Frequency	2.4 GHz
Path Loss Index	3
Add. Losses	35 dB
Gain TX Antenna	1.76
Gain RX Antenna	1.76
Noise Figure	10

References

[1] A. Chandrakasan et al. Power aware wireless microsensor systems. *ESSCIRC*, September 2002.

[2] M. Gastpar and M. Vetterli. On the capacity of wireless networks: The relay case. *Proc. IEEE INFOCOM*, 3:1577–1586, June 2002.

[3] H. Hashemi. The indoor propagation channel. *Proceedings of the IEEE*, 81:943–968, July 1993.

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

[5] P. Herhold, E. Zimmermann, and G. Fettweis. A simple cooperative extension to wireless relaying. *Int. Zurich Seminar on Communications*, February 2004.

[6] R. Min. *Energy and Quality Scalable Wireless Communication*. PhD thesis, Massachusetts Institute of Technology, 2003.

[7] K. Schwieger, H. Nuszkowski, and G. Fettweis. Analysis of node energy consumption in sensor networks. *European Workshop on Sensor Networks*, January 2004.

[8] W. Ye, J. Heidemann, and D. Estrin. An energy-efficient MAC protocol for wireless sensor networks. *Proc. IEEE INFOCOM*, 2002.

Drahtlose Sensornetzwerke: Aspekte der Hardwareminiaturisierung

David-Dmitry Polityko, Michael Niedermayer, Stephan Guttowski, Werner John, Herbert Reichl

Fraunhofer Institut für Zuverlässigkeit und Mikrointegration (IZM)
Gustav-Meyer-Allee 25 / D-13355 Berlin

Die aus einer Vielzahl an Sensorknoten bestehenden drahtlosen Netzwerke gelten als Vorläufer der zukünftigen Anwendungsformen von Informations- und Kommunikationstechnologien, die durch Miniaturisierung und Einbettung von Mikroelektronik in andere Objekte sowie ihre Vernetzung und Allgegenwart im Alltag gekennzeichnet ist. Die Vision der allgegenwärtigen, in der Umgebung verteilten autarken Mikrosysteme – häufig als „Pervasive/Ubiquitous Computing, Smart Dust, oder auch eGrain" bezeichnet - setzt einen enormen Fortschritt in der Hardware-Miniaturisierung voraus. Viele heutige prototypische Umsetzungen von Sensorknoten für drahtlose Sensornetzwerke, die auf kommerziell verfügbare diskrete Bauelemente und Standardfertigungsprozesse wie z.B. SMT zurückgreifen, erreichen bereits einen hohen Miniaturisierungsgrad. Die Potenziale der modernen Technologien der Aufbau- und Verbindungstechnik werden dabei jedoch meist nicht ausgereizt. Um zukünftig an die äußerste Grenze der Miniaturisierung zu gelangen, sind neue Ansätze in der Systemintegration notwendig. Im Rahmen des Forschungsprojekts „AVM – Autarke Verteilte Mikrosysteme" ist am Fraunhofer Institut für Zuverlässigkeit und Mikrointegration ein Sensorfunkmodul für Technologietestzwecke entstanden. In diesem Beitrag werden am Beispiel des Sensorfunkmoduls bzw. -netzwerkknotens, die Miniaturisierungspotenziale von Hardware diskutiert, die beim Übergang von der reinen Surface Mounting Technology (SMT) zur ungehäuste Halbleiter verwendenden Schaltung (MCM Multichip Module) entstehen. Darüber hinaus werden die damit verbundenen technologischen und entwurfstechnischen Aspekte dargestellt. Anschließend wird auf weitere die Miniaturisierung der autarken verteilten Mikrosysteme bestimmende Faktoren eingegangen und ein Ausblick auf die geplanten Forschungsarbeiten in diesem Bereich gegeben.

1. Einführung

Verglichen mit der noch vor 10 Jahren weit verbreiteten Durchstecktechnik für die Montage von diskreten Bauelementen, erlaubt die heutige Oberflächenmontage-Technologie (SMT Surface Mounting Technology) kombiniert mit der modernen Leiterplattentechnik eine beachtliche Ersparnis bezüglich den Abmessungen der Baugruppe. Die im Rahmen des 1. Fachgesprächs sowie in der Fachpresse vielfach präsentierten Hardwareplattformen für Sensornetzwerke, die sich dieser Techniken bedienen, können als eine Illustration dienen [5,6].

Ein erheblicher Anteil der Schaltungsfläche wird i. d. R. durch integrierte Schaltkreise (IC Integrated Circuits) belegt. Betrachtet man die so genannte Siliziumeffizienz (Silicon Efficiency) – Verhältnis aus der eigentlichen Halbleiterfläche eines integrierten Schaltkreises zu der Gesamtfläche, die durch sein Gehäuse und der Kontaktierung auf der Leiterplatte belegt wird [2] – so stellt man fest, dass insbesondere hier im Bereich des Packagings weitere Potenziale zur Miniaturisierung durch die Flächenreduktion vorhanden sind (Abbildung 1).

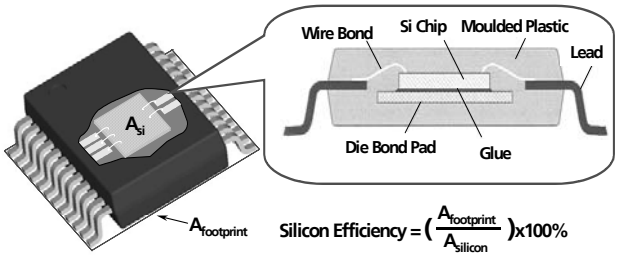

Abbildung 1: Aufbau eines gehäusten SMD IC und Siliziumeffizienz

Ein direkter Einsatz von ungehäusten sogenannten „nackten" Halbleiterchips (Bare Dice) auf dem Schaltungsträger erlaubt bei der Entwicklung von Funksensorknoten ein hohes Maß an Flächenreduktion zu erreichen. Auf diese Weise entstehende miniaturisierte elektronische Mikrobaugruppen werden als Multichip Module (MCM) bezeichnet [1].

In dem BMBF Forschungsprojekt „AVM – Autarke Verteilte Mikrosysteme/eGrain" beschäftigt sich das Fraunhofer Institut für Zuverlässigkeit und Mikrointegration mit technologischen und entwurfsmethodischen Fragestellungen. Im Folgenden diskutierter Sensorfunkknoten dient hierbei als funktionaler Prototyp. Die erste Ausführung in herkömmlicher SMD-Technik dient als Testplattform, anhand derer spezifische Systemkenntnisse und Erfahrungen für die Hardwareintegration gesammelt werden können. In den weiteren Systemminiaturisierungsschritten werden nicht nur technologische Ansätze applikationsbezogen überprüft, sondern auch Rückschlüsse auf eventuelle Änderungen der Systemarchitektur abgeleitet sowie generelle Erkenntnisse über Designmethoden derartiger Systeme ermittelt. Bereits bei der ersten Miniaturisierungsstufe – beim Übergang von SMT zu MCM – wurde demonstriert, dass bei gleich bleibender Funktionalität die für die Elektronik eines Sensorfunkknotens erforderliche Fläche um den Faktor 2,5 bis 5 reduziert werden kann.

2. Sensorfunkmodul-Prototypen

2.1. Hardwarekonzept

Die Prototypen des Sensorfunkmoduls können schaltungstechnisch und räumlich in vier Funktionsböcke unterteilt werden (Abbildung 2):

1. **Sensorbaugruppe,** die neben Licht- und Temperatursensor auch Bedien- und Signalisierungselemente beinhaltet (Schalter, Leuchtdioden zur Kommunikations- und Batteriestatusanzeige);
2. **Mikrocontrollerblock,** der die Licht-, Temperaturmesswerte sowie den Feldstärkeindikator der empfangenen Funksignale über einen integrierten Analog-Digital Umsetzer (ADU) erfasst und die Kommunikation steuert;
3. **Funkschnittstelle**, welche aus einem ISM-868,6MHz Transceiver und einer über die Steckverbinder integrierten Helixantenne besteht;
4. **Energieversorgung,** die mittels einer Lithium-Knopfzelle den Leistungsbedarf für das Gesamtsystems bereitstellt.

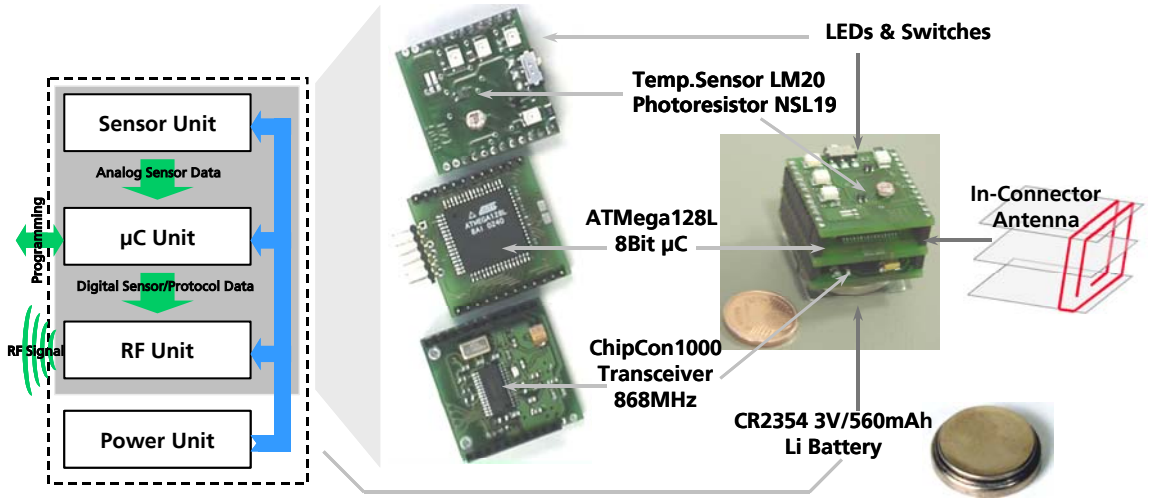

Abbildung 2: Sensorfunkmodul - Hardwarearchitektur und Aufbau des SMD-Prototypen

Der erste Prototyp wurde in herkömmlicher SMD-Technik als ein steckbares System aus 3 Platinen realisiert, die durch Steckverbinder mechanisch und elektrisch zusammengefügt und über einen Batteriehalter montiert werden. Die jeweils 26x26 mm großen Platinen wurden aus herkömmlichem zweilagigem Cu-kaschiertem FR4 Substrat gefertigt und entsprechen den Anforderungen der heutigen Leiterplatten-Standardfertigung (Strukturbreite 200µm, Strukturabstand 150µm, Viadurchmesser 500µm).

2.2. Exemplarische Anwendung

Um die Funktionsfähigkeit aller Hardwarekomponenten zu verifizieren, entstand eine exemplarische Netzwerk-Anwendung (Abbildung 3). Fünf Sensorfunkknoten lassen sich dabei in einem einfachen Master-Slave-Sensornetzwerk betreiben, mit dem die flächige Temperatur- und Lichtverteilung angezeigt werden kann. Die Messwerte werden von jedem Knoten zwei mal pro Sekunde mit 9,6 kBaud manchester kodiert an die weiteren Sensormodule sowie die Basisstation, die in ihrer Architektur den gleichen Mikrocontroller und Funkbaustein enthält, gefunkt. Über eine RS232 Schnittstelle gelangen die Daten von der Basisstation in einen PC, wo sie gespeichert oder in einer grafischen Oberfläche in-situ dargestellt werden können. Bei der Implementierung stand der Nachweis der Funktions- und Netzwerkfähigkeit der Prototypen-Hardware im Vordergrund, und weniger die Entwicklung eines hochdatensicheren Netzwerkprotokolls oder von hochpräziser Messtechnik. Die dafür geschriebene Software ist schichtweise aufgebaut und lässt viele weitere Protokollimplementierungen zu.

Abbildung 3: Verteilte Messung von Licht und Temperatur im Sensornetzwerk mit Basisstation

2.3. MCM-Prototyp

Beim Prototyp der ersten Miniaturisierungsstufe war es das Ziel, die lateralen Abmaße der elektronischen Schaltung soweit wie möglich zu reduzieren. Im Ergebnis konnte die Gesamtfläche der Elektronik von 40cm² auf 16cm² bzw. das Volumen von 16cm³ auf 8cm³ gesenkt werden. Dazu wurden der Mikrocontroller (µC) und der Funkschaltkreis als Flip Chips (FC) aufgebaut. Im Unterschied zur Drahtbondtechnik wird hier der Halbleiterchip mit den Anschlüssen nach unten auf dem Substrat z.B. durch leitfähiges Kleben montiert. Der größte Faktor in der Flächenreduktion von ca. 15 ergibt sich daher beim Vergleich der durch die integrieren Schaltkreise belegten Flächen (Tabelle 2). Dabei benötigt der Mikrocontroller und die gesamte Funkbaugruppe in der Realisierung als Multichip-Modul weniger Fläche als der Mikrocontroller im TQFP-Gehäuse allein (Abbildung 4).

Version	***Components square*** *[mm²]*						***Total Functional Square*** *[mm²]*	***Total Surface*** *[mm²]*
	Sensors	*Passives (RLC)*	*IC*	*LED*	*Quartz*	*Others*		
SMD	19,10	60,64	333,79	43,20	19,80	292,16	768,69	3x52²=4056
MCM	15,51	30,92	22,01	14,28	10,00	155,76	248,48	2x40²=1600
Reduction Factor	1,2	2,0	15,2	3,0	2,0	1,9	3,1	2,5

Tabelle 2: Flächenvergleich

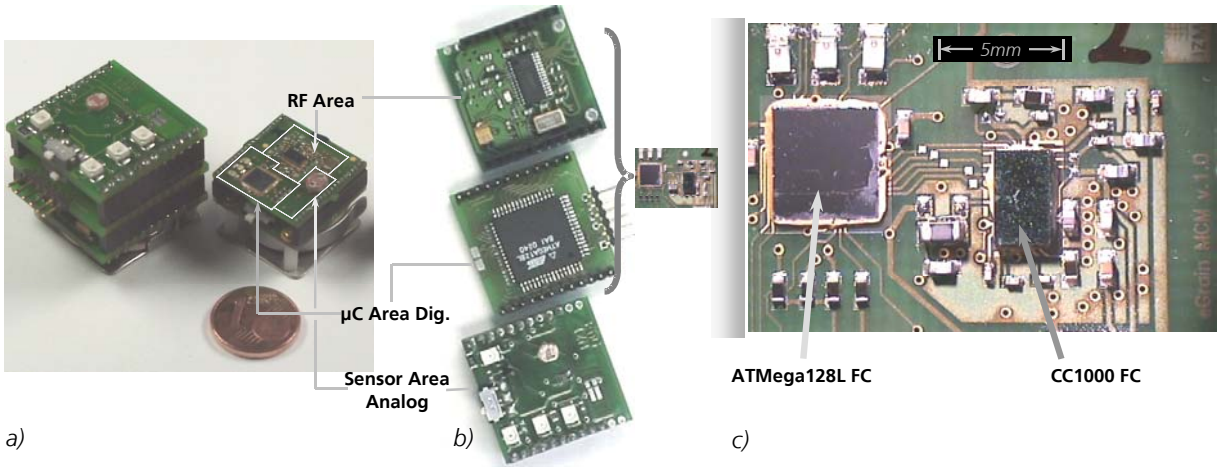

Abbildung 4: a) Direkter Vergleich der SMD und MCM-Versionen; b) Flächenvergleich von Mikrocontroller und Funkbaugruppe; c) Vergrößerte Abbildung von Mikrocontroller und Funkbaustein der MCM-Version

Nicht zuletzt hat die Verwendung der heute kleinsten verfügbaren passiven SMD-Bauelemente (Widerstände, Kapazitäten und Induktivitäten) in der 0201 und 0402 Bauform (gegenüber in der SMD-Version üblichen 0603 und 0805) sowie kleinerer Leuchtdioden (0603 statt PLCC2) zur Miniaturisierung des Sensorfunkknotens beigetragen.

Die aus zweilagigem Cu-FR4-Substrat realisierte Platine fällt nach IPC (Institute for Interconnecting and Packaging Electronic Products) Klassifikation unter die MCM-L (laminated) Kategorie, die alle auf fortgeschrittener Leiterplattentechnologie (wie HDI - Hign Density Interconnects Technologie) basierenden Module zusammenfasst. Dies ist - verglichen mit MCM-C (Keramiksubstrate/Dickfilm/Hybridtechnik) und MCM-D (Substrate aus Halbleiterfertigung/Dünnfilm) - wohl die einfachste, aber auch i. d. R. die preiswerteste MCM-Technologie. Das Modul gehört mit einer Strukturbreite von 75µm und kleinstem Strukturabstand von 50µm zu den anspruchvollen aber realisierbaren Leiterplattenanfertigungen.

Berücksichtigt man die Tatsache, dass nur die obere 20x20 mm große Platine beinahe die gesamte Schaltung beinhaltet, so verbessert sich der Reduktionsfaktor für die Gesamtfläche von 2,5 auf 5, was die Miniaturisierungspotenziale der Elektronik durch die Verwendung von Bare Dice generell sowie durch FC-Technik im konkreten Beispiel des Funksensorknotens hervorhebt.

Die MCM-Technologie weist viele Vorteile auf: Verringerung von Größe und Gewicht, Verkürzung der elektrischen Verbindungen bzw. Signallaufzeiten innerhalb einer Baugruppe, Verringerung einiger parasitärer Effekte, leichtere Abschirmung eines kleinen Systems gegenüber äußeren Einflüssen etc.[1,2]. Sie erfordert aber auch höhere Kosten im Entwurf und in der Herstellung. Die einem ASIC Entwurf ähnelnde Entwicklung eines MCM unterscheidet sich stark von der Entwicklung einer Standardbaugruppe und erfordert von dem MCM-Designer tiefergehende Kenntnisse der Aufbau- und Verbindungstechnik. Bei der Herstellung der MCM werden in der Regel hochwertige Substrate mit einer

hohen Packungsdichte feiner Verdrahtungsstrukturen benötigt, was zusammen mit dem Aufwand für Handling und Prozessierung der Bare Dice sich entsprechend in den Kosten auswirkt.

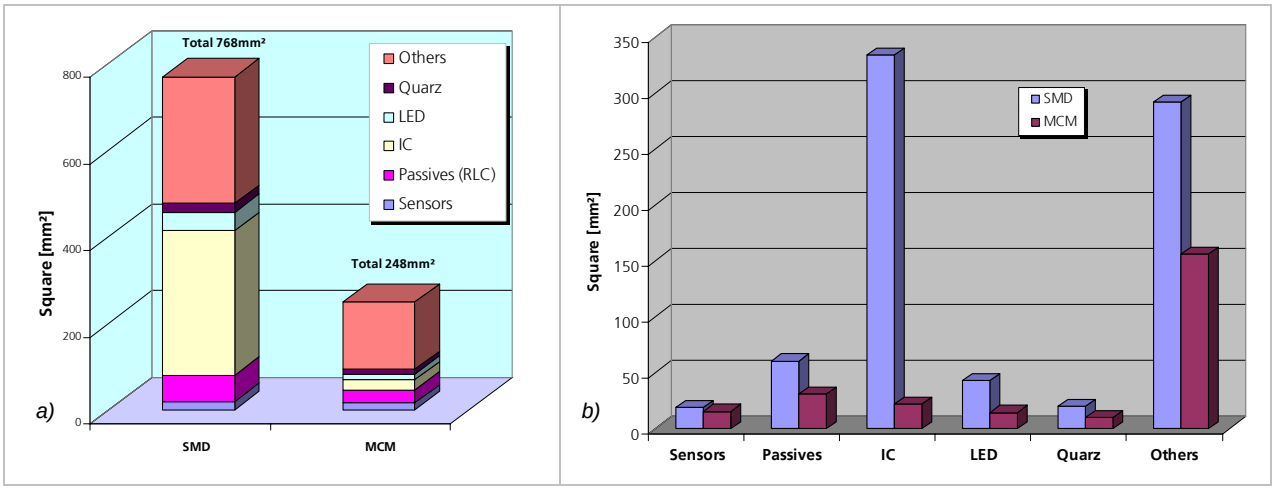

Abbildung 5: a) Vergleich der durch Bauelemente belegten Flächen SMD- vs. MCM-Version b) Gegenüberstellung einzelner Komponentengruppen

3. Komponentenorientierte Betrachtung

Eine Betrachtung der einzelnen Komponenten hinsichtlich der belegten Fläche auf dem Schaltungsträger erlaubt gute Rückschlüsse auf weitere Miniaturisierungspotentiale (Abb. 6). Während beim SMD-Prototyp die integrierten Schaltkreise einen Großteil an der belegten Fläche von ca. 43% belegen und damit die Gesamtabmaße erheblich beeinflussen, ist bereits bei der MCM-Variante der Flächenanteil der Halbleiterchips kleiner als die der passiven Außenbeschaltung. Damit kann gezeigt werden, dass die weitere Verkleinerung des Systems nicht allein durch stetigen Fortschritt in der IC-Halbleitertechnologie vollzogen werden kann. Für weitere Untersuchungen zur Systemminiaturisierung müssen die Skalierungspotenziale aller Komponentengruppen neu bewertet werden:

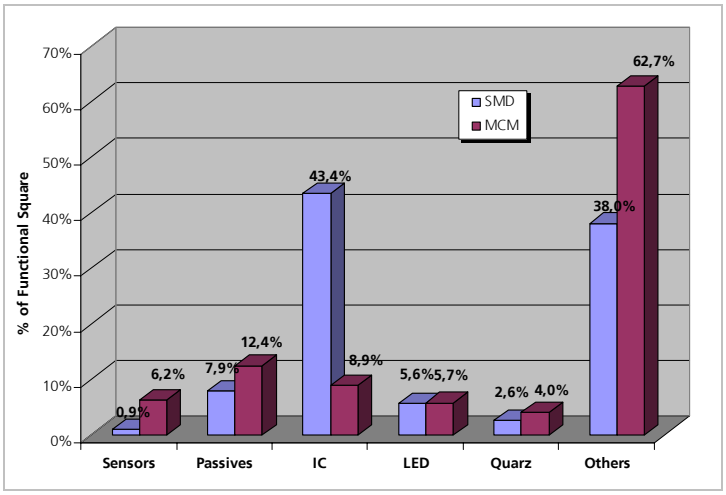

Abbildung 6: Prozentanteile der Komponentengruppen an der belegten Fläche

3.1. Sensoren

Die hier verwendeten Sensoren haben nur einen exemplarischen Charakter. Eine Aussage über Größenreduktion hängt stark von der Art der Sensorik ab. Während einige temperatursensitive Elemente als eine ultradünne unpassivierte Schicht abgeschieden werden können, erfordern MEMS-Beschleunigungssensoren oftmals eine hermetische Verkapselung. Im Falle der hier verwendeten Licht- und Temperatursensoren ist erkennbar, dass mit sinkenden Sensorabmessungen auch die physikalische Ankopplung an das Messmedium geringer wird, so dass eine weitere Verkleinerung auf Kosten der Präzision geschehen kann. Auch nimmt der unerwünschte Einfluss durch umgebende Schaltungselemente zu. So müsste der Lichtdetektor ausreichend weit entfernt von der Leuchtdioden bzw. der Temperatursensor nicht zu nahe am Mikrocontroller platziert werden. Außerdem werden die Anforderungen hinsichtlich Präzision, Bandbreite und Stromverbrauch bei der Miniaturisierung zum Teil recht unterschiedliche Schwerpunkte in der Schaltungsoptimierung setzen.

3.2. Passive Komponenten

Die Gruppe der Widerstände, Kondensatoren und Spulen besitzt ein erhebliches Verkleinerungspotenzial. Es sollten immer so wenig wie möglich diskrete Komponenten genutzt werden, sofern diese Komponenten auch auf dem Siliziumchip zu integrieren sind. Neben integrierten Schaltkreisen mit minimaler Außenbeschaltung können auch passive Netzwerke (passive arrays) die Integrationsdichte erhöhen. Die im MCM-Prototyp verwendete 0201-Bauform (0,6x0,3x0,3mm) ist bereits manuell schwer zu verarbeiten und erreicht mit der nächsten Entwicklungsstufe 01005 (0,3x0,15x0,15mm) beinahe mikroskopische Dimensionen. Ergänzend dazu kann eine Einbettung von Widerständen

und kleinerer Kondensatoren und Spulen in den Schaltungsträger (Embedded Passives) eine weitere Miniaturisierung ermöglichen.

3.3. Integrierte Schaltkreise

Die Geometrien der Halbleiterchips - IC-Flächen wurden innerhalb von 10 Jahren um den Faktor 40 verkleinert [4]. Die von Moore postulierte kontinuierliche Verdoppelung der Strukturdichte erreichte ein Stadium, in dem die Sprünge in der Halbierung der absoluten Flächen immer kleiner werden (Abbildung 7). An Stelle einer massiven Flächenverkleinerung könnte man durchaus eine Komplexitätsteigerung der Chips erwarten. Es sind bereits Chips auf dem Markt, die in CMOS-Technik monolithisch integrierte HF-Transceiver und µC-Kerne vereinen, und damit eine weitere Option für die Verkleinerung des IC-Anteils im System aufzeigen.

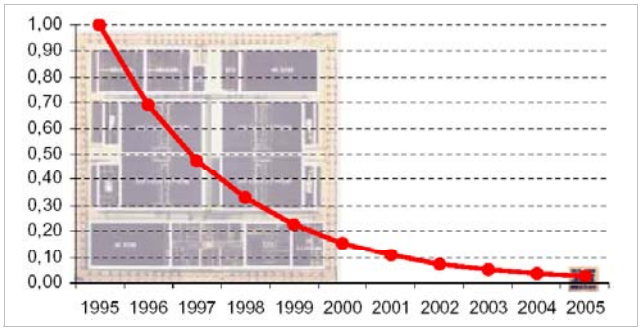

Abbildung 7:Entwicklung der Chipflächen durch Verringerung der Strukturbreiten [4]

3.4. Quarze

Ein Großteil der per Funk kommunizierenden Systeme benötigt eine präzise Taktgebung, die in ausreichender Genauigkeit und Stabilität von Quarzen zur Verfügung gestellt werden kann. Die Grenzen der Miniaturisierung dieser Komponenten sind klar gesetzt: Heute verfügbare Quarzoszillatoren mit Gehäuseabmessungen von wenigen mm befinden sich fast in der Größenordnung eines Quarzkristalls. Anders als bei passiven Komponenten sind bislang keine praktikablen Lösungen zum Einbetten des Quarzes in ein Substrat bekannt. Daher muss das System so ausgelegt werden, dass möglichst wenig Quarze benötigt werden (z.B. Verzicht eines Quarzes für den Echtzeittakt).

3.5. Leuchtdioden und Sonstige Bauelemente

Neben den Leuchtdioden wurden unter den „sonstigen“ Bauelementen alle die Komponenten zusammengefasst, die die Funktionalität nur indirekt beeinflussen (wie Programmierstecker, mechanischer Schalter, Steckverbinder, etc.) und mit zunehmender Miniaturisierung ein erhebliches Hindernis darstellen werden. Diese Komponenten sollten entweder komplett vermieden oder durch neuartige Lösungen ersetzt werden. Der Programmspeicher kann beispielsweise entweder auf der Waferebene oder im Laufe der Modulfertigung mit einem Bootloader beschrieben werden, der eine spätere Programmierung über Funk ermöglicht.

4. Weitere Miniaturisierungsaspekte

Neben einer Größenreduktion durch Anwendung moderner Technologien der Aufbau- und Verbindungstechnik sind auch einige limitierende physikalische Faktoren zu beachten. So wird die parasitäre Kopplung dicht positionierter Bauelemente und Zuleitungen oftmals nicht mehr zu vernachlässigen sein. Zwei weitere Komponenten sind insbesondere hinsichtlich des Volumens zu betrachten: nämlich die Antenne und die Energieversorgung, welche schwer ohne Herabsetzung der Reichweite bzw. Lebensdauer zu miniaturisieren sind.

4.1. Antenne

Die kleinstmöglichen Abmessungen der Antenne, stehen im direkten Zusammenhang mit der Sendefrequenz bzw. Wellenlänge des Funksignals. Ein Wechsel in die höheren Frequenzbänder wie 24 GHz erlaubt zwar kleinere Antennenabmessungen, jedoch muss die daraus folgende stärkere Übertragungsstreckendämpfung der Funksignale über höhere Sendeleistungen ausgeglichen werden. Auch größere dielektrische Verluste werden die Leistungsaufnahme im HF-Frontend ansteigen lassen, so dass die Volumenersparnisse im Antennenbereich teilweise zu deutlich größeren Energieversorgungen führen. Als Gegenmaßnahme können jedoch höhere Datenraten realisiert werden, wodurch über kürzere Betriebszyklen zur Übertragung der Datenpakete der Gesamtstromverbrauch sogar sinken kann. Eine schnellere Architektur führt zu höheren Aufwänden in der Synchronisation und teilweise völlig anderen Schwerpunkten einer Schaltungsoptimierung, so dass die Wahl des Frequenzbandes hinsichtlich kleinster Abmaße nicht einfach zu treffen ist.
Im Rahmen der Forschungsprojektes „Autarke Verteilte Mikrosysteme“ soll auch ein 24GHz-Frontend mit einer auf dem Chip integrierten Schlitzantenne entwickelt werden. Die Richtwirkung dieser Antenne wird eine parallele Kommunikation zweier benachbarter Sensorknotenpaare erlauben, um dabei die Vorteile einer räumlich minimierten Überlappung der Kommunikationskanäle in ihrer Wirkung auf die Netzwerkfunktionalität zu untersuchen.

4.2. Energieversorgung

Im Gegensatz zu massiven Verkleinerungen in der Halbleitertechnologie, erhöhte sich die Energiekapazität von elektrochemischen Energiequellen in den letzten 20 Jahren lediglich um 20% [3]. Bei kleiner werdenden Abmessungen steigt der Volumenanteil des Gehäuse bzw. der Verkapselung, was die theoretisch erreichbaren Werte für die Energiedichte nach unten korrigiert. Eine maximal mögliche Leistungsoptimierung aller Systembestandteile sollte immer angestrebt werden. Ein Durchbruch könnte langfristig durch Mikrogeneratoren, die andere Prinzipien zur Energiewandlung (wie regenerative Quellen in der Umgebung, Mikrobrennstoffzelle etc.) nutzen, herbeigeführt werden. Je nach Anwendung können drahtlose Aufladetechniken die Lebensdauer heraufsetzen.
In der Energiebilanz von drahtlosen Sensornetzen benötigt meist die drahtlose Kommunikation per Funk zur Netzwerkadministration und zum Weiterleiten von Daten den größten Anteil im Leistungsbedarf. Daher wird oftmals eine Arbeitsteilung zu prüfen sein, die neben winzigen Sensoren mit minimalen Kommunikationsmöglichkeiten auch größere leistungsfähigen Netzwerkknoten bereithält, um nicht mehr auf eine geringe Netzwerkfunktionalität begrenzt zu sein.

5. Zusammenfassung und Ausblick

Die hier vorgestellten Arbeiten sind die ersten Bestandteile einer Miniaturisierungs-Roadmap, die im Projekt „Autarke Verteilte Mikrosysteme“ vom Fraunhofer Institut für Zuverlässigkeit und Mikrointegration angestrebt wird (Abbildung 8). Neben dem Nachweis der Potenziale der MCM- bzw. FC-Technik für die Miniaturisierung von Hardware für drahtlose Sensornetzwerke, brachte die experimentelle Entwicklung bereits im ersten Schritt wichtige Erfahrungen über die bestimmenden Faktoren beim Entwurf autarker verteilter Mikrosysteme.

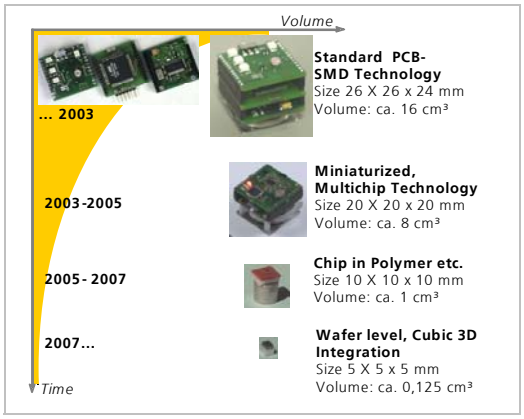

Abbildung 8: Design und Technologie Roadmap

Aus der Betrachtung der funktionalen Flächenanteile folgt, dass die zweidimensionale Systemintegration sicherlich in ihren Möglichkeiten noch nicht vollständig ausgeschöpft ist. Dennoch sind die Grenzen der weiteren Miniaturisierung greifbar. Die Erschließung der dritten Dimension – nicht mehr planare sondern vertikale, dreidimensionale Systemintegration – eröffnet neue Horizonte für zukünftige Entwicklungen [2,7]. Die Erforschung der damit verbundenen technologischen Aspekte (faltbare Schaltungsträger, Montage und Einbettung von ultradünnen ICs, vertikale Kontaktierungsmöglichkeiten etc.), aber auch der systemtechnischen Herausforderungen (zunehmende Überkopplung der Komponenten, thermische Wechselwirkungen, Erarbeitung einer Entwurfsmethodik für 3D-Systemintegration) gehört zu den aktuellen Arbeitsthemen des Fraunhofer Institut für Zuverlässigkeit und Mikrointegration.

6. Literatur

1. Chandler, N. et al.: „Multichip Module Design handbook, Edition 2“, Europractice MCM Service, 1999
2. Al-sarawi, et al.: „A Review of 3-D Packaging Technology“ 2 IEEE Transactions On Components, Packaging And Manufacturing Technology—Part B, Vol. 21, No. 1, February 1998
3. Hilty, L. et al.: „Das Vorsorgeprinzip in der Informationsgesellschaft. Auswirkungen des Pervasive Computing auf Gesundheit und Umwelt”, Zentrums für Technologiefolgen-Abschätzung, TA-SWISS, Bern, 1999
4. Reichl, H.; Wolf, J.: „Trends in der Aufbau- und Verbindungstechnik“, 5. Erlanger Seminar LEF, Erlangen, 6./7.3.2002
5. Lifton, J. et al.: „Pushpin Computing System Overview: A Platform for Distributed, Embedded, Ubiquitous Sensor Networks“, in Mattern, M.; Naghshineh, M. (Eds.): „Pervasive Computing, First International Conference, Pervasive 2002“, Proceedings, LNCS 2414, pp. 139–151, Zürich, 2002
6. Barton, J. et al. „Miniaturised Modular Wireless Sensor Networks“ NMRC Ireland, MIT Cambridge, 2002
7. Reichl, H.: „eGrain – elektronischer Staub“, in Fraunhofer Magazin, 4/2001

Die Inhalte dieser Publikation sind ein Teil der Arbeitsergebnisse des Projektes „AVM – Autarke Verteilte Mikrosysteme/eGrain“. Das wissenschaftliche Vorprojekt „AVM/eGrain“ wird durch das Bundesministerium für Bildung und Forschung gefördert (Förderungskennzeichen: 16SV1658). Gesamtziel des wissenschaftlichen Vorprojektes „AVM/eGrain“ ist die Vorbereitung zukünftiger Miniaturisierungstechnologien auf mehreren Ebenen (Software, Hardware, Energietechnik, Kommunikation, etc). Für die Inhalte dieser Publikation sind ausschließlich die Autoren verantwortlich.

Künftige Standards für Sensor-Aktor-Netze im „intelligenten Gebäude“

Klaus Kabitzsch, Mario Neugebauer
TU Dresden, Fakultät Informatik

Kurzfassung

Der Beitrag berichtet über laufende Standardisierungsaktivitäten der Europäischen Union in den oberen Schichten von Sensor-Aktor-Netzen für das Gebäudemanagement.

1. Einleitung

Die Entwicklung drahtloser Sensornetze ist mit der Vision von Tausenden bzw. Zehntausenden von Knoten verbunden, die jeweils als Teil eines großen Systems an einer gemeinsamen Aufgabe mitwirken. Dabei konzentrieren sich die Arbeiten derzeit auf die unteren Schichten. Demgegenüber finden andere, ebenso wichtige Themen nur wenig Beachtung:

- Reale Anwendungen der o.g. Größenordnung sind bisher kaum implementiert, weshalb praktische Erfahrungen zum notwendigen Engineering, zu sinnvollen Tools usw. nicht existieren und daher die Inspiration für entsprechende Arbeiten fehlt.
- Als applikative Aufgabenstellung wird meist ein simples Monitoring einfacher, gleichförmiger Daten und deren Konzentration in einer zentralen Datenbank angenommen. Reaktive Systemfunktionen (vom Sensor zum Aktor) werden nur selten betrachtet. Eine zentralistische Implementierung solcher Funktionen würde meist an Performance-Problemen scheitern, müsste also lokal und autonom in den einzelnen Knoten geschehen.
- Wenig Beachtung wird der Frage geschenkt, wer später mit welchen Methoden die Funktionen dieser großen Systeme (und damit der vielen Tausend einzelnen Knoten) festlegen soll. Eine klassische Programmierung jedes einzelnen Knotens ist ökonomisch ausgeschlossen. Bei vielen der angedachten Anwendungsvisionen ist der Einsatz von Fachpersonal (Informatiker) zur Errichtung sogar grundsätzlich zu teuer.

2. Stand bei drahtgebundenen Sensor-Aktor-Netzen

Oft wird übersehen, dass andere Domänen bereits über entsprechende Erfahrungen und Lösungsansätze verfügen. Für die o.g. Sachverhalte trifft dies z. B. auf die Gebäudesystemtechnik zu, welche seit über zehn Jahren vernetzte, „intelligente“ Gebäude praktisch implementiert. Allerdings sind die dortigen Netze nicht auf drahtlose Übertragung festgelegt, sondern nutzen einen Mix aus drahtgebundenen Techniken, Power Line und Funk je nach praktischer Opportunität.

- Inzwischen setzt man dort Schaltkreise ein, die auf einem Chip den Verarbeitungsprozessor sowie zwei Kommunikationsprozessoren (für alle 7 OSI-Schichten) vereinen. Bis heute sind über 40 Mio. dieser Chips in vernetzte Gebäudesysteme implementiert worden, wo sie sowohl komplexere Funktionen als auch einfachste Aufgaben wie die von Lichtschaltern, Türschlössern usw. übernehmen.
- Gebäude mit Netzen, in denen über 20.000 solcher Knoten in einer Gesamtanlage zusammenwirken, sind inzwischen keine Seltenheit mehr. Im Gegensatz zu anderen Bereichen der Netztechnologie hat Europa hier eine Vorreiterposition inne. Zur „Programmierung“ dieser Systeme sind Hochschulabsolventen bzw. entsprechende Informatik-Kenntnisse nicht mehr notwendig. Weiterhin erforderlich sind dagegen die Fachkenntnisse der entsprechenden Bau-Fachdisziplinen. Allerdings sind aus Kostengründen je Knoten insgesamt nur wenige Minuten Engineering-Zeit zulässig.

3. Entwurf von Sensor-Aktor-Netzen für Gebäude

Inzwischen haben sich gesicherte Erfahrungen zu sinnvollen Architekturen, arbeitsteiligen Vorgehensweisen bei Entwurf, Inbetriebnahme und Wartung, zu Tools usw. etabliert. Diese werden derzeit aufbereitet, um sie in Europäischen Normen festzuschreiben. Die TU Dresden ist in diese Arbeiten integriert, die im CEN TC247WG4 bzw. WG5 stattfinden. Die angestrebten Konzepte sind durch folgende Grundzüge gekennzeichnet:

- Für den Entwurf der Netzwerktopologie, die Auswahl der Geräte (Knoten) aus dem weltweiten Angebot, die Festlegung der Adressen und Kommunikationsparameter sowie die Übertragung des Entwurfs in das reale Netz nutzt man grafische Tools.
- Die Programmierung ähnelt dem Vorgehen bei JavaBeans und geschieht ebenfalls durch grafische Tools. Sie beginnt mit der Selektion vorgefertigter Softwareobjekte und deren logischen Zuordnung zu den Knoten. Danach wird die Kommunikation zwischen diesen durch grafische „Bindings“ editiert. Am Ende wird der Entwurf durch Download im Netz verteilt und die Inbetriebnahme des Gesamtsystems geeignet unterstützt.

4. Standardisierungskonzepte auf Schicht 7

Die beschriebene Vorgehensweise beim Entwurf und Aufbau der vernetzten Systeme kann funktionelle Vielfalt nur dann kostengünstig realisieren, wenn bei der Auswahl der Geräte (Knoten) und Software-Objekte im Tool auf den weltweiten Vorrat an Hard- und Software-Produkten zurückgegriffen werden kann. Dieser wird derzeit auf ca. 7000 unterschiedliche Typen geschätzt. Eine freie Auswahl und Kombination aus diesem Gerätefundus ist aber nur möglich, wenn gewisse Voraussetzungen der wechselseitigen Passfähigkeit auf funktioneller Ebene erfüllt sind. Zu diesem Zweck wird ein hierarchisches System von Standards vorgeschlagen, welches diese Interoperabilität sichert, andererseits aber die Freiheit der Hersteller nicht zu stark einschränkt, sich am Markt voneinander zu differenzieren:

- Die Kommunikation auf Schicht 7 erfolgt durch Austausch von „Netzwerkvariablen“, deren Datenfluss ereignisorientiert organisiert ist und so die Scheduler der Empfänger-Chips anspricht. Die Konstrukte des speziellen, im Chip genutzten C-Dialekts, Adressierung, Netzwerkmanagement usw. sind auf dieses Konzept angestimmt.
- Da Kommunikation nur über „Bindings“ zwischen typgleichen Netzwerkvariablen möglich ist, sind für alle relevanten Aufgaben „Standard-Netzwerk-Variablen-Typen“ (SNVT’s) definiert. Nach diesem Standard gibt es also weltweit nur eine einzige zulässige Art, wie man Messgrößen (Feuchte, Druck, Temperatur), Betriebszustände, Alarmmeldungen usw. über das Netz überträgt.
- Für typische Anwendungsfunktionen (Sensoren, Regler, Aktoren usw.) sind Standard-Software-Objekte definiert, deren grundsätzliche Verhaltenseigenschaften von allen Komponenten-Herstellern eingehalten werden müssen.
- Für die Software-Objekte der einzelnen Branchen (Heizung, Beleuchtung, Personenzutritt, Jalousien usw.) gelten noch spezielle Festlegungen, welche in Profilen festgelegt sind.

5. Quasistandards für Tools

Nutzt man ein Tool zum Gesamtentwurf eines großen Netzes und der in ihm implementierten Software-Funktionalität, so entsteht eine umfangreiche Datenbasis, in der alle notwendigen Entwurfsentscheidungen gespeichert sind. Aus dieser Datenbank wird bei der Einrichtung und Inbetriebnahme der Download in das Netz gespeist, später aber auch Diagnose, Wartung und Rekonstruktion unterstützt. Auch die Leitrechner nutzen das in der Datenbank abgelegte Architekturwissen, um Beobachtungs- und Bedienzugriffe des Menschen von der Leitebene in die Peripherie zu ermöglichen. Für diese Datenbank-Struktur hat sich ebenfalls ein Quasistandard entwickelt, so dass in den aufeinanderfolgenden Entwurfsphasen unterschiedliche Tools jeweils auf

die Zwischenergebnisse ihrer Vorgänger aufbauen können. Solange diese Datenbank auch zur Laufzeit am Netz verbleibt, sind Wartungsarbeiten und sonstige Änderungen am Netz jederzeit möglich, ohne dass die Konsistenz zwischen realer Anlage und Entwurfsdaten verloren geht. Dies gilt auch bei zeitlich parallelen Eingriffen mehrerer Personen unabhängig voneinander (Fernwartung).

6. Literatur

Loy, D.; Dietrich, D.; Schweinzer, H. (Eds.): Open Control Networks. , Kluwer Academic Publishers Boston, Dordrecht, London 2001

Kabitzsch, K.; Dietrich, D.; Pratl, G.: Gewerkeübergreifende Systeme.
VDE Verlag Berlin Offenbach 2002

Stein, G.; Kabitzsch, K.: A Radio Protocol for Low Power Wireless Sensor Networks.
Proc. FeT2003 5th IFAC International Conference on Fieldbus Systems and their Applications, Aveiro, Portugal, July 2003, pp. 269 - 274 (IFAC International Federation of Automatic Control)

Neugebauer, M.; Plönnigs, J.; Kabitzsch, K.: Prediction of Network Load in Building Automation.
Proc. FeT2003 5th IFAC International Conference on Fieldbus Systems and their Applications, Aveiro, Portugal, July 2003, pp 53 - 58

Stein, G.; Kabitzsch, K.: Concept of an Architecture for Wireless Building Automation.
Proc. 6th IEEE AFRICON Conference, George, South Africa, Sept. 2002, Vol. 1, pp 139 - 142

Kabitzsch, K.; Stein, G.: Network Profiles for LON. Proceedings 4th FeT'2001, IFAC International Conference on Fieldbus Systems and their Applications, Nancy, France, Nov. 2001, pp. 231 – 234

Rauscher, T.; Fischer, P.; Kabitzsch, K.: Ein branchenübergreifendes Klassenkonzept für Feldbusobjekte.
It+ti Informationstechnik und Technische Informatik 42 (2000) H. 4, pp. 38 – 44, Oldenbourg Verlag München

Kabitzsch, K.: LON-Multivendoranlage: Testbett für Feldbussysteme in Industrie und Gebäude.
Automation & Drives Kompendium, S. 59 - 60, KM Verlag München 1998,

BlueBeacon, persönlicher E-Commerce mit Bluetooth

Dr. Till Harbaum, BeeCon GmbH, Haid-und-Neu-Strasse 7, 76131 Karlsruhe, Germany
EMail: harbaum@beecon.de, Phone: +49 721 4998963, Fax: +49 721 4998962

1 Die Idee

Die Bluetooth-Funktechnik unterscheidet sich von allen bekannten anderen Datenfunktechniken (WLAN, Zigbee, ESM-Systeme, ...) vor allem dadurch, dass über die reine Funktechnik hinaus sogenannte Anwendungsprofile unterstützt werden. Diese Profile erlauben es unter anderem, Datenobjekte zwischen einer vielzahl verschiedener Geräte auszutauschen, ohne dass dazu Software installiert werden muss, oft ist dazu nichteinmal der Eingriff des Anwenders nötig.

Die Zahl der mit der Bluetooth-Funktechnik ausgerüsteten Geräte steigt beständig.

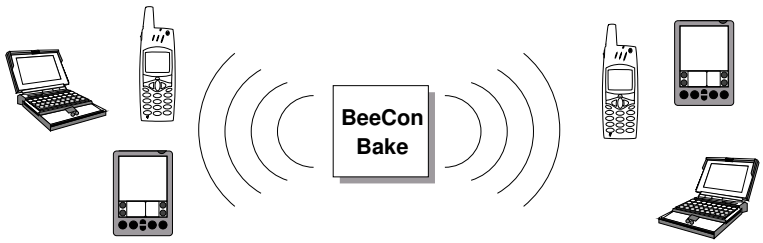

Abbildung 1: Objekt-Aussendung mit BlueBeacon

Zu den Objekten, die sich mit dieser Technik an PDAs, Mobiltelefone und Notebooks austauschen lassen gehören:

- Kontakte/Visitenkarten/Adressbucheinträge, diese können personenbezogene Daten, aber auch Firmenadressen, -telefonnummern und Internetadressen enthalten, so dass ein Empfänger die übertragenen Daten direkt in sein Adressbuch übernehmen kann.
- Termine- und Aufgaben werden in Geräteinterne Kalender und Aufgabenlisten übernommen. Übertragen lassen sich zum Beispiel Messetermine sowie Einträge zu Produktvorstellungen, Aktionen auf Messeständen etc.
- Bilder- und Tondokumente werden je nach Zielgerät in passende Anwendungen integriert. Die Darstellungs- und abspielmöglichkeiten sind häufig geräteabhängig und der Sender muss ggf. über entsprechende Informationen über das Zielsystem verfügen.
- Programmdaten erlauben schliesslich die Übertragung von beliebigen ausführbaren Programmen auf das Zielsystem. Die Spanne möglicher Anwendungen reicht von Werbespielen bis zu konkreten Anwendungen wie Fahrplansystemen etc. Hierzu ist die exakte Bestimmung des Zielsystemes nötig.

2 Plattformerkennung

Während des Datenaustausches wird unter anderem eine eindeutige Gerätekennung übertragen. Diese Kennung läßt sich auf mehrere Arten nutzen.

Die Gerätekennung beihaltet eine Herstellerkennung, über die sich der Hersteller des angesprochenen Gerätes feststellen läßt. Bei Herstellern, die nur eine begrenzte Produktpallete (zum Beispiel Nokia) anbieten läßt sich daraus oft schon ableiten, um welches konkrete Gerät es sich handelt.

In einem zweiten Schritt können von jeden erreichbaren Gerät angebote Dienste abgefragt werden. Die verfügbaren Dienste erlauben weitere Rückschlüsse auf die Geräteart, so dass sich zum Beispiel PDAs von Mobiltelefonen unterscheiden lassen.

Zusätzlich kann aus Kommunikationsparametern oft auf konkrete Softwareimplementierungen geschlossen werden, so dass sich sogar einzelne Betriebssysteme und spezielle Geräte erkennen lassen.

Zusammengenommen erlauben diese Parameter ein Zielgerät-orientierte Aussendung von Daten. Zum einen lassen sich zum Beispiel Video- und Audioclips exakt an das addressierte Zielsystem anpassen, Bilder in einer zum Zielsystem passenden Auflösung und Audioclips in passender Qualität übertragen. Zusätzlich lassen sich Programmdaten speziell auf die Zielplattform zugeschnitten aussenden, um zum Beispiel Spiele und andere Anwendungen in der für jede angestrebte Zielplattform nötigen Form zu übertragen.

3 eCoupons

Ein anderer vielversprechender Einsatz der eindeutigen Gerätekennung findet sich in der Erstellung elektronischer Coupons. Es lassen sich an das konkrete Zielsystem gebundene Daten in Form von zum Beispiel eCoupon-Addressbucheinträgen erzeugen. Diese eCoupons werden für jedes angesprochene Geräte automatisch mit einer gerätespezifischen Kennung versehen, so dass jedes angesprochene Gerät einen einmaligen eCoupon erhält.

Das Einlösen eines Coupons erfolgt im einfachsten Fall durch die Rückübertragung an Systeme wie zum Beispiel den BeeCon MicroBAP (Internetanbindung) oder den BlueSwitch (Coupongesteuertes aktivieren von Geräten). Sobald ein Kunde einen per BlueBeacon erhaltenen Coupon durch Zurücksenden an eine elektronische Couponbox einlöst läßt sich feststellen, ob der Coupon für dieses System ausgestellt wurde. Der Kunden kann den Coupon nicht weitergeben bzw. ein zwischenzeitlich über ein drittes Gerät übertragener Coupon kann als solcher erkannt werden und zum Beispiel zurückgewiesen oder dem ursprünglichen Empfänger gutgeschrieben werden.

4 Realisierung

Der BlueBeacon-Code läßt sich auf den meisten BeeCon-Produkten nutzen. Der in Abbildung 2 dargestellte BlueMP3 ist nur ein Beispiel dafür. Der BlueMP3 ist in der Lage, 20 Stunden aus zwei AA-Mignon-Zellen bis zu 20 Stunden lang aktiv nach Kommunikationspartnern zu suchen und Datenobjekte zu übertragen. Das Geräte erfüllt diese Aufgabe vollständig wartungsfrei und ohne jeden Benutzereingriff.

Ein speziell auf die Aufgabe als BlueBeacon zugeschnittenes Gerät ist noch deutlich kleiner, als dieses System. Alternativ kann auch eine Hardware mit zusätzlichen Funktionen einsetzen, so ist zum Beispiel die MicroBAP-Hardware mit einer zusätzlichen Ethernet-Schnittstelle ausgerüstet, mit der sich Informationen über kontaktierte Geräte sofort ins Internet einspeisen lassen.

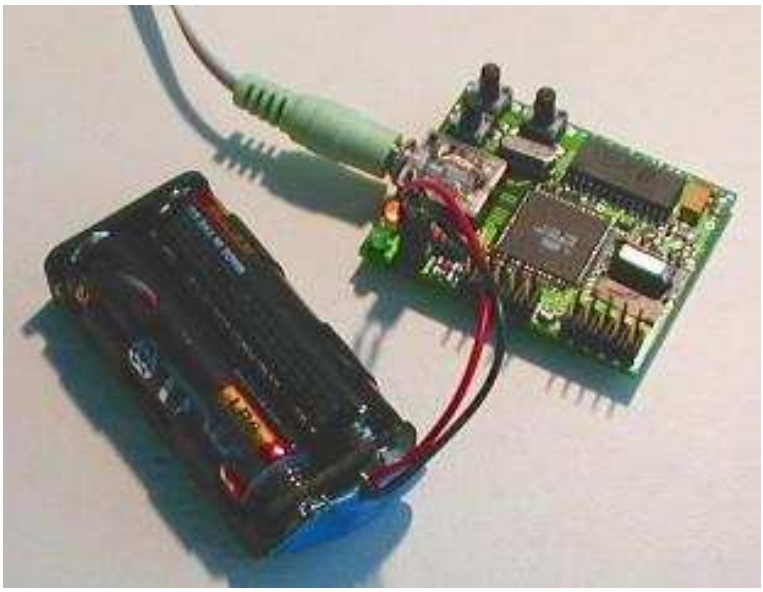

Abbildung 2: Der BlueMP3 mit Batterien

5 Einsatzszenarien

Zu den vielen denkbaren Szenarien zählen unter anderem:

- Der BlueBeacon als mobiler Werbeträger, am Körper getragen, in ein Fahrzeug eingebaut oder stationär eingesetzt erlaubt die Übertragung einfacher Werbebotschaften oder Firmen- und Kontaktadressen. Zum Beispiel in einem Taxi eingesetzt kann ein Kunde mit der Telefonnummer des Taxiunternehmens vesorgt werden. Durch die direkte Übertragung der Kontaktinformation in das Mobiltelefon des Kunden ist es diesem bequem möglich, das Taxiunternehmen bei weiterem Bedarf erneut zu erreichen.

- Der BlueBeacon im Messeeinsatz erlaubt die Übertragung von Messestand-Standorten ("Besuchen sie uns Halle X, Stand Y") oder die Verbreitung der Termine für Show-Events durch Kalendereinträge ("Produktepräsentation XYZ um 15 Uhr Halle X Stand Y").

- Ein BlueBeacon eCoupon-System besteht aus zwei Bestandteilen: Dem BlueBeacon-basiertem eCoupon-Generator, der an nahezu beliebigem Standort platzierbar ist und dem System, bei dem die eCoupons eingelöst werden können. Beispielanwendung ist wieder die Messe, auf der abseits des eigenen Messestandes Coupons verteilt werden, die dann am eigenen Messestand eingelöst werden können.

Damit sind die Einsatzmäglichkeiten des BlueBeacon-Systems bei weitem nicht erschöpft. Der geringe Preis, der geringe Platz- und Energiebedarf und nicht zuletzt die Wartungsfreiheit dieser Systeme eröffnen Szenarien, die mit üblicher PC-Technik nicht erreichbar wären.

6 Größe der potenziellen Zielgruppe

Mit entsprechend ausgerüsteten Geräten (PDAs, Mobiltelefonen) läßt sich die Erreichbarkeit anderer Geräte manuell testen. Einfache Versuche haben ergeben, dass die Durchdringung mit entsprechenden Geräten inzwischen in einem typischen Eisenbahnwagon, Kinosaal oder Kantine/Mensa ausreicht, dass sich dort in der Regel ein Kommunikationspartner findet. Bei von entsprechendem Publikum besuchten Plätzen (Messen, Tagungen, 1.ter Klasse Eisenbahn, Flugzeug/Flughafen) ist die Zahl der potentiellen

Kommunikationspartner deutlich höher, wobei das so erreichtbare Publikum oft auch die angestrebte Zielgruppe darstellt.

Die Erreichbarkeit ist stark Geräteherstelleranhängig. Die Erfahrungen zeigen, dass ca. 90% der erreichbaren Geräte Nokia-Mobiltelefone sind, die übrigen Geräte sind PalmOS und Pocket-PC-basierte PDAs sowie Mobiltelefone anderer Hersteller. Oftmals geht die Erkennbarkeit mit der Bedienung der Geräte einher, so sind PalmOS-basierte PDAs immer dann erkennbar, wenn sie eingeschaltet sind.

Security for Sensor Networks

Matthias Gerlach, Fraunhofer FOKUS (gerlach@fokus.fraunhofer.de)

March 9, 2004

1 Introduction

Sensor networks are foreseen to become an integral part of our everyday life in the future. This is due to their envisioned small node size, cost and manifold sensing capabilities. Acceptance of and trust in this new technology will strongly depend on the security provided for sensor networks.
Despite its importance, security for sensor networks has gained only little attention in the last few years. A couple of security suites for sensor networks have been published(e.g. [7, 11, 3, 9]).
This paper provides a definition of security requirements and an overview of existing approaches. The provided paper is *work in progress*.
First, the term sensor networks and its building blocks are defined to allow for a classification of the approaches with respect to the kind of sensor networks addressed. Then, a definition of security for sensor networks is given and different security classes are discussed. Further, existing security suites for sensor networks are compared and discussed with respect to the sensor networks they address and their security provided. Finally, future work in the field of security for sensor networks is identified.

1.1 Sensor Network Definition

There are several different views of what a sensor network is. Naturally, these different views result in different approaches to design a security suite. For the purpose of this work first a generic definition is provided.
A sensor network typically consist of a high number of resource constrained sensor nodes, and one or several base stations. Further, the network contains aggregation points which may be implemented on all or only on dedicated sensor nodes.
The actual data gained from the sensor nodes is available via the *base station*. In addition, the base station can be seen as a bridge between the sensor network and external networks, e.g. IP networks. Multiple base stations may be interconnected via existing communication infrastructure. The base station is assumed to have laptop class computing power.
The sensor network core consists of many[1] *sensor nodes* distributed randomly over a certain area. These nodes are low energy, low cost devices. They incorporate a microprocessor with RAM (and ROM), a radio transceiver, some sensor/actuator electronics and an ADC/DAC, all assembled in a compact package. A common example is the MICA mote (cf. `www.xbow.com`). Its main data are depicted in Table 1. The lifetime requirement of a sensor network typically is in the order of months or years. To meet this requirement, a node is put into sleep mode most of the time. This introduces synchronisation issues and thus further complicates the sensor network protocol

Processor	ATMEL ATmega 128L
Processor speed	4MHz
Flash	128KB
SRAM	4 KB
EEPROM	4 KB
Instruction Set	8 bit
External Power	3 V
Processor Current Draw	5.5 mA, $< 20\mu$A in sleep mode
Radio Current Draw	12mA transmit, 1.8 mA receive, $< 1\mu$A in sleep mode
Power Supply	2x AA batteries (approx. 6600 mWh)

Table 1: Mica Mote Data Overview

[1] In the order of thousands, possibly millions.

design.
Aggregation points may be seen as an abstract construct. They can either be implemented as dedicated aggregation nodes, or just be code which may be executed on any node on demand. Aggregation points fuse data to reduce redundant data transmission.
The communication patterns in sensor networks depend on the application. Usually, the nodes communicate locally, and only significant events are sent via multiple hops to the base station. On the other hand the base station sends configuration requests to all associated nodes. Other authors propose different communication paradigms for different applications, such as direct communication with the base station [11] .

1.2 Security Requirements

Security for networks commonly comprises six major aspects. These are

- Authentication,
- Integrity,
- Confidentiality,
- Freshness,
- Availability,
- Nonrepudiation.

In this section, these aspects are defined and discussed with respect to their relevance in sensor networks.

1.2.1 Authentication

From the message point of view authentication ensures that a message can be unambiguously correlated to a sender. No impersonation of a valid identity by an adversary shall be possible.
As with localisation in large scale sensor networks, there may not be unique ID's available for authentication purposes. The implications and possible solutions are considered future work. In general, in sensor networks, it is more important to authenticate the application rather than the node.

1.2.2 Integrity

The integrity requirement in sensor network shall ensure that messages traverse the network unaltered. Integrity usually inherently comes along with encryption. Integrity will be proved when the encrypted message can be decoded. This proof assumes that an intruder cannot decipher, alter and then encipher the message again.
For applications where no confidentiality needs to be provided, an unforgeable hash checksum is sufficient. In practise, a MAC[2] is transmitted with the message to prove its integrity and authenticity. Even if authentication and integrity proof often go along, it should be noted that they are different requirements. However, authentication without integrity is useless and vice versa.

1.2.3 Confidentiality

Wireless networks use a broadcast medium where an eavesdropper remains undetected as long as she does not launch an active attack, i.e. actively tries to break into the network and alter information. Dependent on the level of security to be obtained, messages need to be encrypted to provide confidentiality for the data sent within the network.
This may have several reasons. First, encrypted messages prevent an adversary from obtaining routing information. Note that this is not entirely true. By correlating transmit events, a route may be discovered. Consider the case that a node far away from the base station sends a packet. An adversary with a sensitive receiver being able to locate radio transmissions may track down the base station or, similarly important, the nodes of the network's backbone. In addition even the fact that a station sends data can disclose the data's contents.
Second, by encrypting messages, the content is kept secret in order not to disclose any information to eavesdroppers.
Considering the strength of the encryption functions, the time horizon for the data need to be considered. An intuitive classification could be

[2]Message Authentication Code

- Short, e.g. just a few seconds[3]. This would be used for information which can be disclosed shortly after they have been sent.
- Medium (Application Lifetime), i.e. if the network is used for different applications during its lifetime, the information needs to stay confidential until the application has finished.
- Network lifetime. This refers to the lifetime of the network. As long as the network is deployed and the user obtains information, all these information need to be secret. Network lifetime for sensor networks is usually in the order of months or years, depending on the applications.
- Forever. Forever describes the fact that data need to remain confident significantly longer than the lifetime of the network. This could be the case in military applications of sensor networks.

Considering the design of encryption algorithms a variety of attacks has to be considered. A typical example is semantic security which shall ensure that the same plain-text is enciphered differently every time [7].

1.2.4 Freshness

In a sensor network, the data obtained shall provide an up-to date and homogeneous view of the network's data. Therefore it must be ensured that data are recent on the one hand and that there are no data duplicates on the other. From the security perspective there need to be mechanisms to prevent replay-attacks, where messages are recorded and replayed by the adversary.

1.2.5 Availability

Availability is not commonly considered as a security requirement. It rather aims at secure network *operation* than secure network *use*. As long as an intruder targets user data, availability is not an issue. But if an intruder for example alters routing messages, the network may not work properly which is indeed a security issue. Karlof et al. describe some attacks on the availability of sensor networks in [4]. In particular in sensor networks, attacks to exhaust the batteries (i.e. *sleep deprivation*[10]), to create routing holes, or to segment the network can impair the network's availability.

1.2.6 Nonrepudiation

Nonrepudiation is especially important in the context of electronic contracting. Nonrepudiation ensures that a sender cannot deny sending a message later. It is usually neglected in the context of sensor networks.

1.2.7 Discussion

Authentication and integrity are the most important aspects for sensor networks. They are often provided at the same time and rely on each other. Depending on the application of a sensor network, confidentiality in the different strengths and availability might become an issue as well. Freshness can generally be considered as being provided automatically, as long as there are no significant latencies within network processing or communication. If replay attacks are to be avoided, clearly freshness is an issue. Availability is an issue in particular concerning routing in sensor networks, as Karlof et al. state in [4]. Nonrepudiation does not matter in sensor networks.
Security is often provided by using cryptographic functions such as encryption. Often, to provide authenticity and integrity, an encrypted hash of the sender's ID, the message content, and a message ID[4], is sent. To additionally provide confidentiality, the whole message may be encrypted. The latter approach is computationally more expensive. The following aspects need to be considered when designing security suites for sensor networks:

1. The level of security to be provided.
2. The prevailing type(s) of communication and, based on these premises:

[3] Or epochs: in TinyOS an epoch is the interval between two queries.

[4] MAC(ID,Data_hash,MSG_ID)

3. the cryptographic functions and protocols to be used,

4. how the keys are distributed. This is called the bootstrapping problem.

Especially the bootstrapping problem can be tedious. Imagine the keys are hard-coded into the nodes on the one hand: an intruder could do anything with the network once she obtains the key. On the other hand, it is problematic to distribute keys to all sensor nodes without using public key mechanisms. These are generally considered too expensive for sensor networks [1].

2 Previous Work

Few proposals have yet been published to establish a security suite for sensor networks which tackle all security challenges. As one of the first existing papers Chen et al. propose a suite of protocols to encrypt the transmission between a node and the base station in [2]. Their approach assumes only node-to-base station communication. It should be noted that this is on of the first papers providing a holistic approach.
To the best of my knowledge, there are five widely known security suites which cover all aspects of security for sensor networks: these are a completely decentralised approach by Slijepcevic et al. [9], a centralised approaches by Perrig et al. [7], Undercoffer et al. [11] and Chen et al. [2]. In addition there is the TinyOS inherent suite developed by Karlof et al..
The security suites are depicted schematically in Figures 2, 4, 5 and 3. A detailed description of these security suites has been left out due to space constraints. The Figures depict the assumed communication type, secure communication channels and key distribution. Figure 1 provides a legend to the security suite schematics.

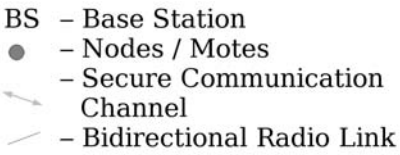

Figure 1: Legend for the Security Suite Figures

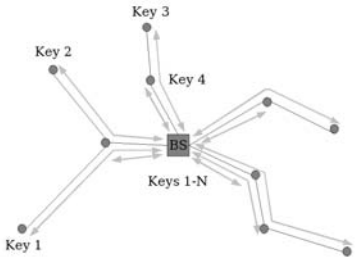

Figure 2: Chen et al.'s and Perrig et al.'s Security Suite. The two suites are based on centralised communication patterns, predeployed keys for unicast communication, and an authenticated broadcast mechanism (TESLA and μTESLA, respectively).

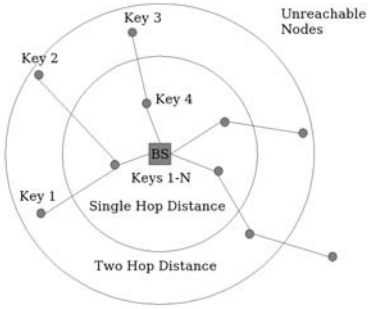

Figure 3: Undercoffer et al.'s Security Suite. The suite assumes at maximum two hop communication and polling for route discovery.

Suite	Communication Type	Eval. Platform	Crypt. Functions	Key Distribution	Provides
Chen et al.	Base station centric, multi-hop	Similar to MICA motes	RC5	Predeployed shared secret key between each node and the base station.	Authentication, Integrity, Confidentialiy (optional)
Perrig et al.	Base station centric, multi-hop	MICA motes	RC5 in CTR mode	Predeployed shared secret key between node and base station.	Authentication, Integrity, Confidentialiy (optional), Freshness
Slijepcevic et al.	Decentralised, multi-hop	WINS nodes	RC6, MD5 hash	Globally shared set of secret master keys.	Authentication, Integrity, Confidentialiy (optional)
Undercoffer et al.	Base station centric, two hops at most	SensorSim, MICA motes	Ciphertext Auto Key (CTAK)	Predeployed shared secret key between each node and the base station.	Authentication, Integrity, Confidentialiy (optional), Availability
TinySec	Base station centric, multi-hop	MICA motes	RC5	Predeployed globally shared secret key.	Authentication, Integrity, Confidentialiy (optional)

Table 2: Comparison of the Different Security Suites

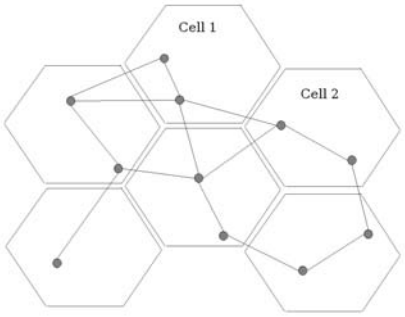

Figure 4: Slijepcevic et al.'s Security Suite. The Figure depicts the security setup for normal network operation, where nodes are associated to cells. Their key is selected with respect to cell membership.

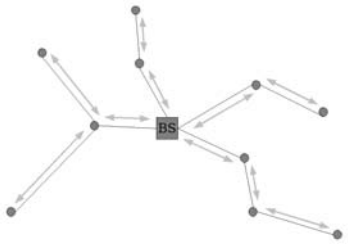

Figure 5: TinySec, the TinyOS Security Framework

2.1 Comparison

A comparison of the aforementioned security suites is provided in Table 2. All authors except Slijepcevic et al. assume a base station centric model and target the MICA platform. As stated above, the base station centric model assumes a laptop class base station. Undercoffer et al. further restrict the routing radius to two hops at most. Thus, their proposal may be less suitable for large scale sensor networks.

All authors assume predeployed symmetric secret keys. TinySec as the simplest approach uses one key for all nodes and the base station. The other approaches differ on the derivation of the actually used key from their so called master keys. Slijepcevic et al. even use a location based procedure to derive keys for node-to-node communication.

The cryptographic functions used are based on the RC5 or RC6 stream ciphers with the exception of Undercoffer et al. which use CTAC, a self synchronising stream cipher. All suites except Slijepcevic et al. use 64 bit keys. Slijepcevic et al. target the more powerful WINS nodes. They use 128 bit keys. A detailed comparison of the used cryptographic functions and their security is future work. As a starting point, Law et al. [5] provide a comparison of relevant ciphers for the EYES platform (cf.

eyes.eu.org).
TinySec, Spins (Perrig et al.) and Chen et al. propose a security suite for a base station centric multi hop sensor network.The propositions of Chen et al. and Perrig et al. do not differ significantly. Due to the shared secret key for each node-to-base station link, local processing will be difficult or at least expensive to set up in the latter proposals. In particular for large scale networks, this will be a significant drawback. However, the authenticated streaming broadcast mechanism introduced by Perrig et al. provides a scalable mechanism to broadcast authenticated information. They have even provided an authenticated routing algorithm based on this mechanism in [7]. μTesla is suitable even for large scale sensor networks.
TinySec on the other hand can only provide weak security and will break down completely if only one node is compromised. Contrarily, it seems to be the best approach when it comes to scalability.

3 Conclusion and Future Work

As can be deduced from the comparison provided, there is yet no sufficiently scalable security suite for sensor networks.
Future work will have to address developing a truly scalable approach to provide security for sensor networks.
Further, scalable and practical mechanisms to exclude malicious nodes from the networks like proposed by Slijepcevic have to be developed for low performance platforms such as the Mica motes. Existing solutions have to be examined with respect to their applicability in large scale sensor networks.
In addition, different cryptographic functions need to be compared and an appropriate one shall be selected. Even if the authors of the mentioned security suites provide a brief statement of other existing and evaluated cryptographic functions, there seems to be some disagreement on the best cryptographic function to be selected. Law et al., e.g., propose different algorithms from the ones used in SPINS and by Slijepcevic et al. claiming their decision to be better.
Last bit not least, authentication mechanisms for large scale sensor networks are yet to be developed. This is due to the fact that in large scale networks, IDs will not be unique anymore. Thus the current ID based authentication schemes will not work.
In a nutshell, there is yet no solution to tackle all security issues in large scale sensor networks. Especially solutions with low communication overhead and suitability for in network processing are not yet available. This leaves room for interesting and challenging research.

References

[1] David W. Carman, Peter S. Kruus, and Brian J. Matt. Constraints and approaches for distributed sensor network security (final). Technical report, NAI Labs, September 2000.

[2] Mike Chen, Weidong Cui, Victor Wen, and Alec Woo. Security and deployment issues in a sensor network, 2000.

[3] Chris Karlof, Naveen Sastry, Umesh Shankar, and David Wagner. TinySec: TinyOS link layer security proposal - version 1.0, July 2002.

[4] Chris Karlof and David Wagner. Secure routing in wireless sensor networks: Attacks and countermeasures, 2003.

[5] Yee Wei Law, Jeroen Doumen, and Pieter Hartel. EYEScrym: EnergY-Efficient Sensor CRYptographic Module. wwwhome.cs.utwente.nl/ ywlaw/pub/prorisc2003.pdf.

[6] S. Park, A. Savvides, and M. B. Srivastava. SensorSim: A simulation framework for sensor networks. In *Proceedings of MSWiM 2000*, August 2000.

[7] A. Perrig, R. Szewczyk, V. Wen, D. Cullar, and J. Tygar. SPINS: Security protocols for sensor networks, 2001.

[8] Bruce Schneier. *Applied Cryptography. Protocols, Algorithms, and Source Code in C.* Wiley, second edition, 1996.

[9] Sasha Slijepcevic, Miodrag Potkonjak, Vlasios Tsiatsis, Scott Zimbeck, and Mani B. Srivastava. On communication security in wireless ad-hoc sensor networks. In *11th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pages 139–144, 2002.

[10] Frank Stajano and Ross Anderson. The resurrecting duckling: Security issues for ad-hoc wireless networks. In *Security Protocols, 7th International Workshop Proceedings*, pages 172–194, 1999.

[11] Jeffery Undercoffer, Saskanth Avancha, Anipam Joshi, and John Pinkston. Security for sensor networks, 2002.

S-CAN: Sicheres Overlay für Sensornetze

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

Motivation

Kleinst-Computer begleiten uns heute mehr und mehr im Alltag und in gewöhnlichen Gegenständen: „Wearables“ stecken an oder sogar in unserer Kleidung, Büroräume verwalten sich selbstständig, Helligkeits-Sensoren regeln automatisch die Beleuchtung und Temperatur-Fühler steuern Heizungen. Solche Sensoren können drahtlos miteinander kommunizieren und Ad-Hoc Netze bilden. Problematisch sind allerdings klassische Sicherheits-Anforderungen wie Vertraulichkeit und Authentizität an diese Kleinst-Computer. Ihre Ressourcen sind in Bezug auf Speicher und Rechen-leistung extrem limitiert, ihre Bandbreite beschränkt, asymmetrische Kryptographie teuer und Batteriestrom allgemein kostbar. Traditionelle Sicherheits-Architekturen sind daher in Sensor-Netzen nur schwer einsetzbar. In dieser Arbeit wird ein Protokoll, „S-CAN“ (Secure Content Addressable Network), zum sicheren Aufbau eines Overlay-Netzes für Sensoren präsentiert, das ermöglicht, Dienste von Sensoren sicher anzubieten, sicher aufzufinden und sicher zu verwenden. Dabei muss der Benutzer in der Lage sein, die Sensoren vor ihrem Einsatz zu personalisieren, ein Anwendungs-Bei-spiel wäre die Heim-Automatisierung durch Sensoren.

Das Content Addressable Network ist ein P2P-Overlay-Netz, das beispielsweise in File-Sharing Umgebungen oder zur Verwaltung eines verteilten Verzeichnisses Verwendung finden kann. Am Institut für Telematik wird das CAN[1] als Overlay-Netz auf Sensoren eingesetzt und um das S-CAN Protokoll erweitert, so daß abgesicherte Kommunikation zwischen den einzelnen Sensor-Knoten möglich wird.

Die grundsätzliche Idee von S-CAN besteht aus der induktiven Erweiterung des Netzes um zusätzliche Sensoren. Ist das Netzwerk zu einem bestimmten Zeitpunkt mit gewissen Knoten in einem „sicheren“ Zustand, dann muss nach dem Hinzufügen eines weiteren Knotens dieser Zustand erhalten bleiben. Das als Grundlage verwendete CAN-Overlay-Netzwerk enthält Dienst-Namen (wie „Temperatur von Raum A“) sowie zugehörige Adressen von Dienst-Anbietern, den Sensoren. Die in diesem Beitrag vorgestellten neuen Ideen im Kontext von S-CAN sind die Integration eines *Zeugenverfahrens* sowie eines *Mehrheitsentscheids*. Die Nachbarn des neu ins Netz hinzuzufügenden Knotens beweisen ihre Authentizität über ein Zeugenverfahren, Anfragen nach Diensten werden durch Redundanz und Mehrheitsentscheide abgesichert.

Überblick: Content Addressable Network (CAN)

Ein CAN ist eine auf alle am Netz teilnehmenden Knoten verteilte Hash-Tabelle. Jeder Knoten verwaltet eine sog. Zone, einen Teilbereich des kompletten Hash-Wertebereichs einer

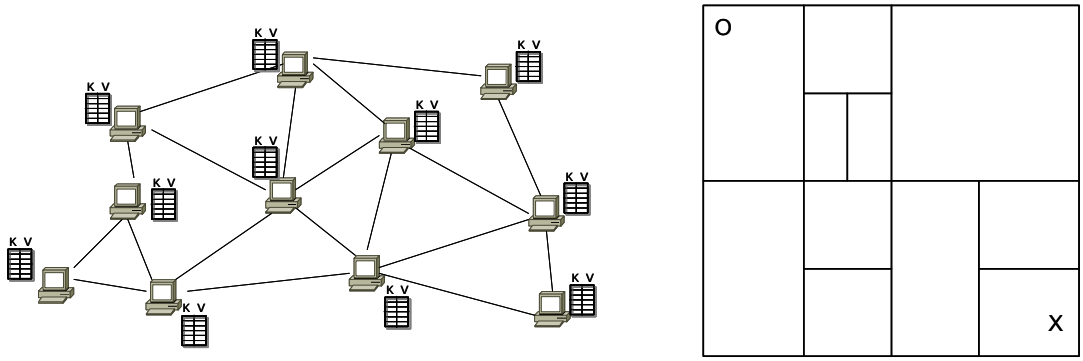

Abbildung 1: a) Verteilte Hash-Tabelle, b) 2-dimensionales CAN

Hash-Funktion *H*. Dies zeigt Abbildung 1a. Die jeweiligen Besitzer einer Zone sind zuständig für das Aufbewahren von (*Schlüssel*, *Wert*)-Tupeln, deren *Schlüssel* in ihrer Zone liegt. Möchte ein Teilnehmer ein (*Schlüssel*, *Wert*)-Tupel im CAN ablegen, so kontaktiert er denjenigen Zonen-Ver-walter, der für die Zone, in dem *H*(*Schlüssel*) liegt, verantwortlich ist. CAN beinhaltet ein Routing-Verfahren, das diese Kommunikation ermöglicht. Genauso kontaktiert ein Knoten auf der Suche nach dem zum *Schlüssel* passenden *Wert* den Zonen-Verwalter, der *H*(*Schlüssel*) verwaltet.
Die Hash-Wertebereich wird dabei in d-Dimensionen eingeteilt, Abbildung 1b zeigt ein Beispiel für zwei Dimensionen. Knoten o und Knoten x verwalten zwei unterschiedliche Zonen der Hash-Funktion. Sie sind über Nachbarn miteinander verbunden. Benachbarte Knoten im CAN verwalten aneinander anliegende Hash-Zonen. Kommt ein neuer Knoten ins Netz, so wird eine vorhandene Zone in zwei Teile aufgebrochen („*split*“). Der alte Zonen-Verwalter und der neue Knoten verwalten dann jeweils die Hälfte der alten Zone.

Die CAN-Funktionalität wird verwendet, um zu Dienste-Beschreibungen Netzwerk-Adressen abzulegen, beispielsweise IP-Adressen oder für Sensor-Netze geeignetere Bluetooth Scatter-Netz-Adressen. Ein Sensor, der einen bestimmten Dienst anbietet, speichert im CAN unter dem Hash-Wert seiner Dienstebeschreibung seine Netzwerk-Adresse ab.

Konzepte von S-CAN
Das Protokoll S-CAN funktioniert induktiv, ausgehend von einem sicheren Zustand wird ein neuer Knoten in das Netz eingefügt („join“), so daß das Netz nach dem „join“ immer noch sicher ist. Das Protokoll verzichtet dabei auf zeitaufwendige Public-Key Operationen, die auf Ressourcen-schwachen Sensoren sehr teuer sind[3].

Die Grundlage für das sichere „join“ in unserem Protokoll ist das sog. „Master-Device“ (MD). Dieses zustandslose Gerät verwendet der Benutzer ausschließlich dafür, um neue Sensoren bewußt in das Netzwerk hinzuzufügen. Das MD ist physikalisch klein, z.B. ein Teil am Schlüsselanhänger des Benutzers. Das MD ist deshalb zustandslos, damit es der Benutzer bei Beschädigung durch ein äquivalentes Gerät von einem sicheren Aufbewahrungsort ersetzen kann. Will der Benutzer seinem Netz einen neuen Sensor hinzufügen, so wird dieser mittels des MD personalisiert. Dies geschieht durch einen „location limited channel“, einen sicheren Übertragungskanal, beispielsweise physi-kalischer Kontakt. Im Moment des Kontaktes kann das MD Geheimnisse auf dem neuen Sensor sicher abspeichern: jeder Sensor bekommt eine zufällige ID und einen nur vom MD direkt aus der ID ableitbaren symmetrischen Schlüssel S_{ID}. Da kein Zustand auf dem MD vorgehalten werden muß, bedeutet dies niedrige Anforderungen an dessen Hardware, eine Batterie ist nicht zwingend notwendig: Strom kann im Moment der physikalischen Berührung vom Sensor aus übertragen werden.

Nach dem Personalisierungs-Vorgang initiiert das MD nun den „join“-Vorgang. Es wählt eine zufällige Position im CAN-Netz, dessen zugehörige Zone aufge-*splittet* werden soll. Der zugehörige Zonen-Verwalter wird gebeten sich zu authentifizieren. Um Mißbrauch zu verhindern, muß der Zonen-Verwalter Zeugen vorweisen, die dem MD bestätigen, daß der Zonen-Verwalter tatsächlich für die gesuchte Zone zuständig ist. Dies sind üblicherweise die direkten Nachbarn des Zonen-Verwalters. Die Sicherheit der Kommunikation zwischen MD und den Nachbarknoten ist durch ihre gemeinsamen geheimen Schlüssel S_{ID} gewährleistet. Die alte Zone wird aufgeteilt und deren Nachbarn entsprechend über die neue Verwaltung informiert. Zudem werden sicher Schlüssel zwischen den Nachbarn und dem neuen Zonen-Verwalter ausgetauscht. Diese beschützen die direkte Kommunikation zwischen zwei Nachbarn im CAN. Zu diesem Zeitpunkt ist das CAN wieder in einem sicheren Zustand: der neue Knoten ist Teil des Netzes, seine direkten Nachbarn können vertraulich mit ihm kommunizieren. Er ist an einer zufälligen Stelle im Netz Zonen-Verwalter, kein Angreifer hat das beeinflusst.

Es ist allerdings noch ein weiterer Schritt notwendig, um Dienste im Netz sicher abzulegen und sicher danach zu suchen. Zunächst gilt, dass kein Verwalter einer Zone bewußt Änderungen an seinem Datenbestand durchführen kann. Da die Hash-Funktion nicht umkehrbar ist, kann er den einzelnen Hash-Werten und den damit verbundenen Adressen keine Dienst-Namen zuordnen. Problematisch ist jedoch, wenn auf dem Rückweg einer Anfrage das Ergebnis der Anfrage, der *Wert*, verfälscht wird. Dies ist aufgrund fehlender Ende-zu-Ende-Sicherheit prinzipiell allen auf dem Antwort-Pfad liegen-

den Knoten möglich. Ein Angreifer-Knoten könnte das Ergebnis so verändern, dass beispiels-weise seine eigene Adresse in der Antwort enthalten ist. Wir schlagen daher vor, Redundanz im CAN zu erzeugen und darauf basierend Mehrheits-Entscheide durchzuführen. Anstatt einer einzelnen Hash-Funktion *H* sollten zwei (oder mehr) zusätzliche Hash-Funktionen *H'* und *H''* Ver-wendung finden. Ein Sensor, der seine Dienste im CAN Netz propagieren möchte, legt seine Adresse nun nicht nur in der Zone ab, in der *H(Dienst-Name)* liegt, sondern auch in den Zonen, in denen *H'(Dienst-Name)* und *H''(Dienst-Name)* verwaltet werden. Da die einzelnen Hash-Werte auf unterschiedlichen Knoten im CAN verwaltet werden, ist die Wahrscheinlichkeit groß, das nicht alle Wege vom anfragenden Dienst-Nutzer zu den Zonen-Verwaltern über einen bösartigen Knoten laufen. Der Dienst-Nutzer sucht analog nicht nur nach dem *Wert* von *H(Dienst-Name)*, sondern auch nach denen von *H'(Dienst-Name)* und *H''(Dienst-Name)*. Über die ihn erreichenden Antworten führt der Dienst-Nutzer dann einen Mehrheitsentscheid durch und verbindet sich zu dem in den Antworten meistgenannten Knoten oder aber schlägt bei nicht Übereinstimmung aller drei Ergebnisse bei einer zentralen Stelle Alarm.

Prototyp-Implementierung

Am Institut für Telematik werden Sensor-Prototypen („BlueNodes"[2]) basierend auf einem 8-Bit ATMEGA 128 Mikrokontroller mit 4 MHz eingesetzt. Als Hauptspeicher stehen 4 KByte RAM zur Verfügung, die drahtlose Kommunikation geschieht derzeit über Bluetooth.

Das S-CAN Protokoll benötigt auf den eingesetzten Sensoren nicht mehr als 900 Byte Hauptspeicher. Es verzichtet darüber hinaus komplett auf rechenintensive und Batterie-belastende Public-Key Operationen. Zur zeit werden am Institut für Telematik Experimente mit Java IButtons für den Einsatz als Master-Device durchgeführt. Die BlueNodes-Prototypen können über physikalischen Kontakt mit einem IButton Daten austauschen.

Fazit und Ausblick

Das S-CAN Protokoll leistet ein sicheres Anbieten und Abfragen von Diensten in Sensor-Netzen. Dafür wird ein Overlay-Netzwerk aufgebaut, das Dienst-Informationen und dazu passende Adressen von Dienst-Anbietern vorhält. Um Mißbrauch vorzubeugen, müssen die zukünftigen Nachbarn eines neu hinzuzufügenden Knotens über ein Zeugenverfahren beweisen, daß sie bestimmte Eigenschaften innehaben. Die sicherheits-relevanten Daten werden im Netzwerk redundant an verschiedenen Positionen abgelegt und die Antworten auf Dienst-Anfragen durch Mehrheitsentscheide auf Richtigkeit überprüft.

Das Protokoll kann allerdings nur in den Szenarien eingesetzt werden, bei denen der Benutzer die Möglichkeit hat, neue Sensoren vor ihrem Einsatz durch ein Master-Device zu personalisieren. Klassische Beispiele hierfür sind die Verwendung im Home- oder Office-Bereich. In anderen Umgebungen, wie dem Abwerfen vieler Sensoren aus dem Flugzeug heraus über einem Zielgebiet, ist dies nur eingeschränkt möglich.

Zur weiteren Absicherung der Kommunikation im Anschluß an die Dienst-Anfrage, den eigentlichen Nutz-Daten-Austausch, könnte als Teil der Adresse beispielsweise ein Public-Key-Zertifikat des Dienst-Erbringer-Sensors mitgeliefert werden. Inwieweit Zertifikate als gemeinsamer Ankerpunkt in den spontanen und ressourcen-beschränkten Sensor-Netzen sinnvoll sind, ist hingegen noch offen. Dies ist jedoch völlig unabhängig vom hier behandelten Registrieren von Diensten bzw. Abfragen von Diensten im CAN.

Literaturverzeichnis

[1] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp und Scott Schenker: „A scalable content-addressable network", Proceedings of the 2001 conference on applications, technologies, architectures, and protocols for computer communications, 2001

[2] Erik-Oliver Blaß, Hans-Joachim Hof, Bernhard Hurler, Martina Zitterbart: „Erste Erfahrungen mit der Karlsruher Sensornetz-Plattform", GI/ITG KuVS Workshop Sensornetze, Berlin, 2003

[3] Yee Wei Law, Sandro Etalle und Pieter H. Hartel: „Assessing Security In Energy-Efficient Sensor Networks", Small Systems Security, 2003

Consensus in WSN – Identifying critical protocol mechanisms

A. Köpke H. Karl
A. Wolisz
Telecommunication Networks Group, Technische Universität Berlin
Sekr. FT 5, Einsteinufer 25
10587 Berlin, Germany
{koepke, karl, wolisz}@tkn.tu-berlin.de
http://www.tkn.tu-berlin.de

1 Introduction

Consensus is an important problem in distributed systems, such as Wireless Sensor Networks (WSNs). It occurs in many places, for instance in the tedious task of updating the software in the nodes in a WSN installation, but also whenever sensors need to agree on a value or actuators agree on an action.

As an example, consider the case where a WSN has been retrofit to a building to save energy, e.g. by using sensor and actuators to open or close the blinds of a room. In such a system, many decisions depending on e.g. temperature, sun shine and wind are possible. The actuators could for instance agree to close all blinds. Or they could agree to half-close all blinds, or to close blind A, while opening blind B and C.

The dependencies between actuators lead to a strict meaning of decision: once a decision is made, an actuator relies on the fact that the other actuators behave according to the decision – there must be a consensus among the actuators.

The question arises how such a consensus can be reached in an energy-efficient manner in a WSN. In a typical solution, one node would start by proposing a value to all participating nodes. After this multicast it expects an answer from all other nodes back. These answers will usually be correlated in time (all nodes start sending shortly after receiving the proposal). Correlation in time means that the network has to deal with a lot of messages within a short period – in the extreme case this can exceed its short-term capacity. However, in a WSN such correlation can be used to aggregate values, turning the apparent disadvantage into an opportunity.

Consequently, we derive a protocol that solves the consensus problem using aggregation. This use of aggregation considerably improves energy efficiency without influencing the quality of the consensus or increasing the time it takes to reach a decision.

The protocol can be formulated using two kinds of aggregation: concatenation or computation of a single value that summarizes all the information. Concatenation allows to find out which nodes did not answer. This can be used to initiate retransmission requests, hence making the transmission more reliable, but in turn increases the packet size. If only a single value is computed, this is not possible. Still, this

This work has been partially sponsored by the European Commission under the contract IST-2001-34734 – Energy-efficient sensor networks (EYES).

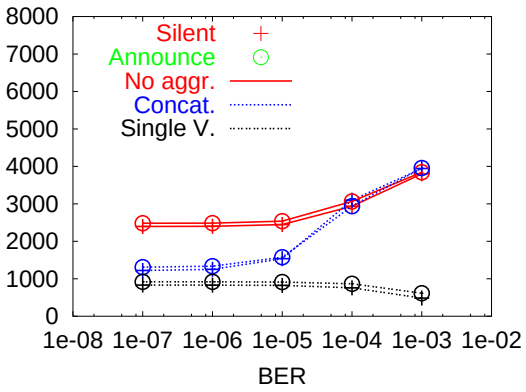

Figure 1: Sent bits per node, 400 nodes, without ARQ

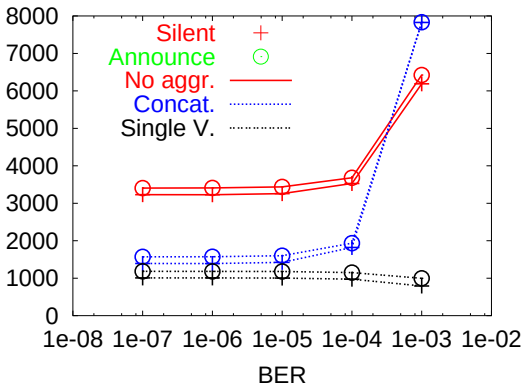

Figure 2: Sent bits per node, 400 nodes, with ARQ

low-cost solution may be sufficient for some applications. In this paper we investigate how reliable the three versions (no aggregation, concatenation, single value) are, measured in the fraction of nodes that correctly decide. We also evaluate how much time these versions need until the protocols terminate and how much energy is spent, measured in the number of bits sent.

2 Selected results

The results presented in this section are averaged for 20 random topologies, simulations were run until a relative precision of 0.01 had been reached (i.e., the actual value is with 95% confidence between $\frac{1}{1.01}\overline{x} \leq \mu \leq 1.01\overline{x}$).

The first metric to look at is the number of sent bits. Fig. 1 and 2 show (with or without ARQ for the unicast communication used to request retransmissions of missing proposal/acknowledgements from individual sensors) the average number of bits sent by each node per consensus attempt as a function of the Bit Error Rate (BER) for a sensor network consisting of 400 nodes, all taking part in the Two-Phase commit (2PC) protocol. The graphs show all the six possible combinations: two ways of building a tree and three forms of value aggregation. The way how a tree is built is denoted by different points in the graphs: if no messages are sent to built the tree, we call this a “silent” tree (the graphs have crosses) whereas when the tree is built using announcements “I am your child” to inform the parent node about the presence, the graphs have circles.

From these figures it becomes clear that how the tree is built has only a minor influence. A more pronounced influence is seen in the way how values are aggregated. The variant that aggregates the values in a single value clearly sends the smallest number of bits and this number even drops with a rising BER. This is due to the fact that with a higher BER more and more messages are lost and not forwarded – resulting in a dropping number of sent bits in downstream nodes. The concatenating variant sends more bits, and this can even exceed the number of bits send by the non-aggregating version – the reason is that the long packets of the concatenating version are more susceptible to losses.

Another question is whether the use of aggregation has an influence on the time it takes for the algorithm to terminate. Figures 3 and 4 show that the use of aggregation does not necessarily increase the time until the protocol terminates. In fact, this is more related to the way timers are chosen for the leafs to start the convergecast, especially for small BER. The problem here is that the possible depth of the tree is vastly overestimated with the maximum number of hops of 30, usually the average number

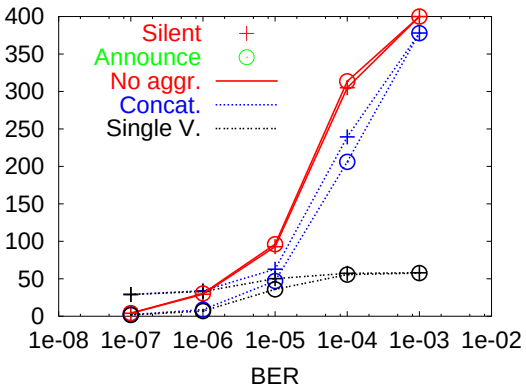

Figure 3: Time to terminate (s), 400 nodes, without ARQ

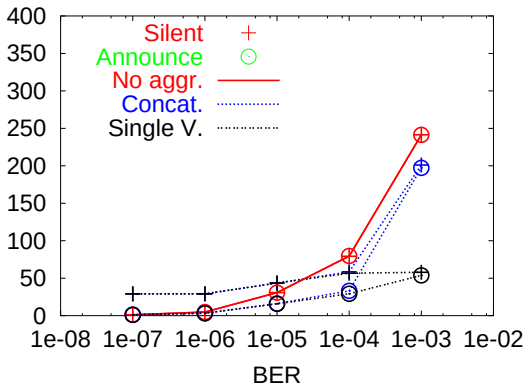

Figure 4: Time to terminate (s), 400 nodes, ARQ

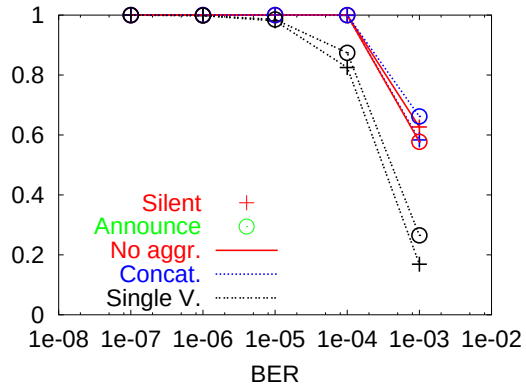

Figure 5: Fraction of decided nodes, 400 nodes, without ARQ

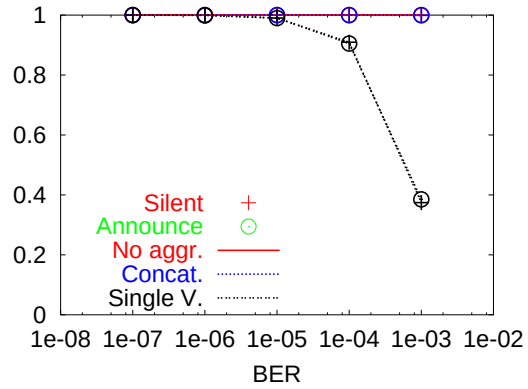

Figure 6: Fraction of decided nodes, 400 nodes, with ARQ

of hops is smaller than 5. This error in the estimated tree depth leads to timeout values (for aggregation and retransmission requests) that are too large. A more sophisticated scheme should be adopted. Still, the variant that does not use aggregation often needs more time – even with the small MAC overhead the large number of messages need time until they are transmitted.

The next (and perhaps most important) question is how aggregation influences the quality of the consensus. The quality of a consensus protocol can be measured in how likely it violates the agreement property. The problem here is that a message might not reach a given node. If a node does not receive the decision, it does not decide. This can happen despite retransmission requests. After three retransmission requests the coordinator terminates the protocol, but some nodes may remain undecided. The fraction of nodes that actually decide gives an impression of the quality of the protocol.[1]

Figures 5 and 6 show that the fraction of decided nodes is different for each variant. The single-value

[1]This is actually a better performance metric than using the more intuitive fraction of incorrectly decided nodes, as this would largely depend on the (relatively small) probability of proposing ABORT and would hence lead to distorted figures – this product of two small probabilities (ABORT probability and the probability that a message of one of the objecting nodes is concerned) is very small and difficult to simulate.

aggregation has the smallest number of decided nodes, because it does not care whether nodes answered or not. The other variants that do care about individual nodes have a very similar performance. The difference between these two variants is not significant up to a BER of 10^{-3}. The difference between the two variants is caused by the way how the tree is built. If the children in the tree do not announce their presence, they will be included in the tree on the first convergecast received from them – this can also be an answer to a retransmission request. If the children have to announce their presence within a certain time limit, then no new children will be included in the tree after that. When the multicast message or their announcement is lost they are not included in the tree – and the next multicast, restricted to the tree, is not forwarded to them. In effect this lowers the number of messages they can potentially receive, lowering the success probability. If the tree is built with more care using Automatic Repeat reQuest (ARQ) the difference diminishes.

3 Future work

The implementation of the 2PC protocol led to some more informal results. Using the simulation we observed that the fraction of decided nodes (the reliability measure) not only depends on the "usual suspects" like number of retransmission requests, reliability of the link layer and reliability of the convergecast structure but also in a subtle way on the timer values.

In general, larger timer values increase the reliability of the protocol but make it more susceptible to errors due to mobility and environmental changes. Furthermore, the increase in the time to terminate may not be acceptable for all applications. A more sophisticated choice of the timer values is thus a natural place to trade off speed versus reliability. Better timer values may also allow to increase the reliability while at the same time lowering the time to terminate. In this sense, the protocols described here are likely to still warrant some further optimizations.

Interestingly, this problem can be formulated in a more abstract way that also implies a possible solution. The number of votes aggregated in a messages constitute the informational value of the message. If a node waits longer it will potentially receive more messages from its children and hence increase the benefit from aggregation. However, waiting is expensive: after some time the gathered information may be outdated or the energy consumed during waiting might be excessive. When should a node stop waiting for answers from its children and send the aggregated value on? This question can be answered using the mathematical theory of optimal stopping. The problem is to find a good stopping rule that can be evaluated on a sensor node.

Partitioning in Mobile Ad Hoc Networks

Jörg Hähner, Dominique Dudkowski, Pedro José Marrón, and Christian Becker
Universität Stuttgart
Institute of Parallel and Distributed Systems (IPVS)
Universitätsstraße 38
70569 Stuttgart, Germany

I. Introduction

Many seminal algorithms for distributed data management used in wired networks assume that communication failures cannot occur [4], and that the only errors nodes may experience are those where a process crashes before the completion of its tasks. Nevertheless, even in the presence of communication failures, the safety property of these algorithms is guaranteed, e.g. a transaction is aborted due to the failure of a communication link. On the other hand, the liveness of those algorithms cannot be guaranteed, e.g. a replicated database does not accept a transaction containing write operations if one of the remote sites is isolated due to communication failure and if strict consistency is applied. In wired networks many innovations have led to big improvements regarding the stability of network connections, so that communication failures are kept to a minimum. This allows classical distributed algorithms to treat these rare occasions as errors from which they can recover after communication is restored.

Wireless networks, especially wireless mobile ad hoc networks, are very different from the perspective of communication failures. On top of the wireless transmission media, which is very error prone, the mobility in such networks leads to a new class of reasons for communication failures. In many application scenarios, mobile devices in such networks are carried by (human) users. In most cases, their movement cannot be influenced by the algorithms running on their devices. Therefore, mobility in general leads to three main types of problems: i) communication links between pairs of devices may be established and lost without notification, yet not every link change leads to a change in partitions, ii) many nodes may be located on a small area resulting in *high* node density, which may lead to the so-called broadcast storms [5], and iii) *low* node density may lead to network partitions where nodes in different partitions cannot communicate with one another. The primary cause for network partitions is of course that of links being established and lost. However, not every link change in the network leads to a change in network partitioning. Both, research on the behavior of individual links and the field of broadcast storms has been a very active field of research in the past years, e.g., [1], [3], [5].

In this paper we present some results describing the partitioning behavior of mobile ad hoc networks.

II. System Model

We assume the system model of a mobile ad hoc network to be comprised of a set of n mobile nodes. Each node is located at a position (x, y) inside of a fixed geographical area A. By $\varrho = n/A$ we denote the average node density in the network. The transmission range of a node, assumed to be equal for all nodes, is defined by the radius r_{tx}, forming a circle whose center is the position of that node. Two nodes n_i, n_j are within transmission range, if the euclidian distance $d(n_i, n_j)$ between n_i and n_j is smaller than r_{tx}.

The *topology graph* $G(N, E)$ consists of a set N of vertices, representing the nodes of the network, and a set E of undirected edges, corresponding to communication links. An undirected edge $\{n_i, n_j\} \in E$ exists, if and only if $d(n_i, n_j) < r_{tx}$. A *network partition* is a subset $P \subseteq N$ where i) a path exists between all pairs of vertices $n_i, n_j \in P$ and ii) no path exists between any pair of vertices $n_i \in P, n_k \in N \backslash P$. By $\mathrm{PART}(G)$ we denote the set of partitions in G.

III. Methodology

In this section we state the motivation and formal definition of the set of metrics used for quantifying the partitioning behavior of mobile ad hoc networks. We focus on metrics that reveal the impact of partition characteristics on data management algorithms. The presented metrics are not completely uncorrelated. Besides characterizing the partitioning behavior of mobile ad hoc networks, one aspect of this work is to find and describe those correlations. To the best of our knowledge, the proposed set of metrics has not been treated in previously published work.

We consider two groups of metrics: *network-wide* partition metrics and *node-centric* partition metrics. The first group addresses the following questions pertaining to the network as a whole: i) What is the number of partitions in the network over time? ii)What is the size of each partition? iii) How frequent do partition splits and joins occur over time? iv) What is the size of two partitions being joined or resulting from a partition split?

Node-centric partition metrics are concerned with the following aspects relating to the perspective of individual nodes: i) How many partition changes does a single node experience over time? ii) How long is the separation time of two nodes located in disjunct partitions? iii) Which nodes are continu-

ously or accumulatively located in the same partition of an individual node over a given time interval?

In this paper we present results for two of the network-wide metrics: the number of partitions and the size of partitions.

A. Preliminaries

Let $T = [0, t_{\max}]$ denote the considered time interval. A *partition event* e occurring at time $t_i \in T$, is defined by a tuple $e = (type, t_i, P_1, P_2, G, G')$. The *type* attribute may be either split or join, indicating that partitions P_1 and P_2 involved in the event are split or joined, respectively. G and G' are the topology graphs before and after the occurrence of the event, respectively. By $e.type$, $e.t_i$, $e.P_1$, $e.P_2$, $e.G$, and $e.G'$ we refer to each element of the tuple e.

We define a partial order $(E_{\text{part}}, <)$, where E_{part} is the set of partition events and "$<$" is the *less-than* relation on their timestamps. The two types of events indicated by the *type* attribute are defined as follows: A *partition join event* $e = (\text{join}, t_i, P_1, P_2, G, G')$, or join event for short, is a partition event in E_{part} transforming a topology graph G into G' such that $P_1, P_2 \in \text{PART}(G)$ and $\exists P \in \text{PART}(G')$ for which $P = P_1 \cup P_2$. A *partition split event* $e = (\text{split}, t_i, P_1, P_2, G, G')$, or split event for short, is a partition event transforming a topology graph G into G' such that $P_1, P_2 \in \text{PART}(G')$ and $\exists P \in \text{PART}(G)$ for which $P = P_1 \cup P_2$. By E_{join} and E_{split}, we denote those subsets of E_{part} that contain only join and split events, respectively.

B. Number of Partitions

Let $\Pi(t_i)$ be the set of partitions that exist at a discrete point in time $t_i \in T$. The *number of partitions* at time t_i in the network is denoted as $|\Pi(t_i)|$. The average number of partitions over T weighted within the timespan between two successive events is calculated as follows:

$$|\Pi|_{\text{avg}} = \frac{\sum_{e \in E_{\text{part}}} |\Pi(t_i)| \cdot \Delta t_i}{t_{\max}} \tag{1}$$

where $\Delta t_i = t_{i+1} - t_i$.

C. Size of Partitions

The *size of a partition* P is denoted as $|P|$. The average size of partitions $\bar{S}(t_i)$ at a given time t_i can be derived from the average number of partitions at t_i by computing $n/|\Pi(t_i)|$. The average size of partitions over T is

$$\bar{S} = \frac{n}{|\Pi|_{\text{avg}}} \tag{2}$$

The number of partitions with size x is calculated as

$$\begin{aligned} H(x) \quad = \quad & |\{P \in \Pi(t_0) \;\mid\; |P| = x\}| + \\ & |\{e \in E_{\text{part}} | e.type = \text{join} \wedge |e.P_1 \cup e.P_2| = x\}| + \\ & |\{e \in E_{\text{part}} | e.type = \text{split} \wedge |e.P_1| = x\}| + \\ & |\{e \in E_{\text{part}} | e.type = \text{split} \wedge |e.P_2| = x\}| \end{aligned} \tag{3}$$

The domain of x is $[1, \ldots, n]$, because the minimum size of a partition is 1 and the maximum size is n. The graph of $H(x)$ is the histogram of the distribution of partition sizes. With $H(x)$ it is possible to identify distributions such as many small and a few large partitions.

IV. Simulation Study

A. System Parameters

The key parameters of mobile ad hoc networks considered when conducting performance evaluation of algorithms are the size of the spatial area in which the mobile nodes may move, the number of nodes in the network, the mobility model the nodes follow, and the transmission range available at the mobile nodes. The size of the spatial area and the number of nodes define the average spatial node density.

We assume that each node in the network is mobile, leading to changes in the topology graph over time. We distinguish between two mobility models: the *random waypoint mobility model* [2] and the *graph-based mobility model* [7].

In the graph-based model, the mobility of nodes is spatially constrained by a graph. Each vertex in the graph represents a particular location (x, y) within a geographic region, such as points of interest, e.g., a subway station in a city center, and road intersections. To connect these locations, edges are implemented in the graph representing routes in the region, such as streets. Each node chooses a vertex of the graph as its destination and a speed from a given interval randomly and moves to that destination on the shortest path of the graph at the chosen speed. In contrast to the random waypoint mobility model, the graph based random walk mobility model is more restrictive and prohibits completely arbitrary movement, reflecting typical scenarios in urban areas.

B. Simulation Results

To conduct our experiments, we have used an event-based simulator which, given a mobility trace, constructs and alters the topology graph over time. The event-based approach is able to capture *every* occurring state of the topology graph over time. The mobility traces were obtained using the simulator presented in [6]. The random waypoint model operates on an area of 2×2 km^2. For the graph based random walk mobility model, we have modelled two graphs, representing a typical Central European city and Midtown Manhattan. Both graphs span a total area of approximately 2.5×1.8 km^2. This allows to compare the simulation results for different mobility models with one another.

1) Number of Partitions: Figure 1 shows the results for the average number of partitions $|\Pi|_{\text{avg}}$ as a function of the number of nodes in the network. For small networks, $|\Pi|_{\text{avg}}$ is mostly determined by the number of nodes in the network, which define the upper bound of the number of partitions. Once the node density is higher than 75 nodes/km^2, which corresponds to 300 nodes in our simulations, $|\Pi|_{\text{avg}}$ steadily decreases. However, even for larger networks with a density of 600 nodes/km^2, i.e. 2400 nodes, the number of partitions may still not be neglected for some classes of algorithms, e.g. implementations of reliable broadcast algorithms which do not take network partitioning into account.

Figure 2 shows an example of $H(x)$ for 400 nodes and the random waypoint mobility model. For the graph based models, we have observed a very similar distribution with respect to the

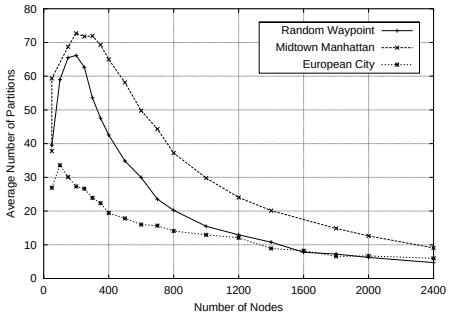

Fig. 1. Average number of partitions $|\Pi|_{\mathrm{avg}}$ over number of nodes

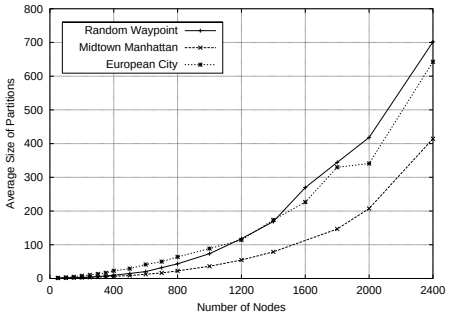

Fig. 3. Average size of partitions $\bar{S}$ over number of nodes

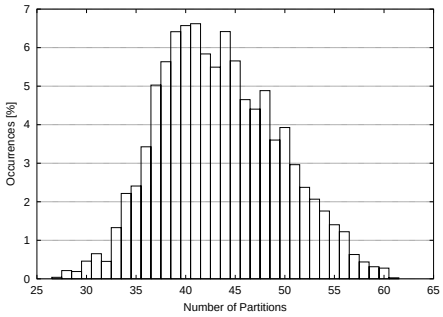

Fig. 2. Histogram for the number of partitions $|\Pi(t_i)|$ for $n = 400$

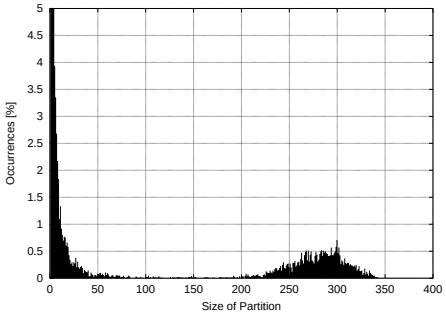

Fig. 4. Histogram for the partition size $H(x)$ for $n = 400$

averages in the number of partitions. With increasing number of nodes, the distribution is even more distinct around the average. Assuming that nodes are able to detect the node density in their vicinity, the selection of a matching distribution allows to reason about the number of partitions in regions of the network for the presented mobility models.

2) Size of Partitions: The results for the average size of partitions $\bar{S}$ as a function of the number of nodes is presented in Figure 3. $\bar{S}$ increases as the number of nodes in the network increases. With increasing size of partitions, the standard deviation of the average partition size increases as well. Figure 4 shows a measurement of the distribution of partitions sizes for 400 nodes, revealing that the occurrence of very small partitions, e.g., isolated nodes and partitions of up to 5 nodes, is very common. In the settings with low density, these small sizes dominate the distribution. With increasing density, the occurrence of very large partitions becomes more frequent and leads to a large standard deviation for the size of partitions.

The result that average sized partitions are rare leads to the conclusion that a node is located in a large partition with high probability, if it is not in a small partition. Detecting whether or not a node is in a small partition is possible with relatively low communication overhead by calculating the k-hop neighborhood, where, for example, $k = 4$ to decide whether or not a node is in a partition with at most 5 nodes. The information on partition size is valuable for many data management algorithms involving, for example, multicasting or broadcasting, since these should be carried out while being in a large partition if a large number of nodes should receive the given message.

REFERENCES

[1] Christian Bettstetter. On the minimum node degree and connectivity of a wireless multihop network. In *Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking and computing*, pages 80–91. ACM Press, 2002.

[2] Josh Broch, David A. Maltz, David B. Johnson, Yih-Chun Hu, and Jorjeta Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*, pages 85–97. ACM Press, 1998.

[3] Michael Gerharz, Christian de Waal, Matthias Frank, and Peter Martini. Link stability in mobile wireless ad hoc networks. In *Proceedings of the 27th Annual IEEE Conference on Local Computer Networks (LCN'02)*, pages 30–39, Tampa, FL, November 2002.

[4] Pnakaj Jalote. *Fault Tolerance in Distributed Systems*. Prentice Hall, 1994.

[5] Sze-Yao Ni, Yu-Chee Tseng, Yuh-Shyan Chen, and Jang-Ping Sheu. The broadcast storm problem in a mobile ad hoc network. In *Proceedings of the Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking*, pages 151–162, Seattle, Washington, USA, August 1999.

[6] Illya Stepanov, Jörg Hähner, Christian Becker, Jing Tian, and Kurt Rothermel. A meta-model and framework for user mobility in mobile networks. In *Proceedings of the IEEE International Conference on Networks (ICON)*, pages 231–238, Sydney, Australia, September 2003.

[7] Jing Tian, Jörg Hähner, Christian Becker, Illya Stepanov, and Kurt Rothermel. Graph-based mobility model for mobile ad hoc network simulation. In *Proc. of 35th An. Simulation Symposium*, pages 337–344. IEEE Computer Society Press, April 2002.

Managing Wireless Sensor Networks

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

1 Introduction

Research in the area of sensor networks is often conducted using simulations. While this is necessary for studies of scalability issues, deploying sensor networks with real hardware helps to uncover problems of the real world and to improve the simulation model. Nevertheless, real sensor networks with 100+ nodes impose new problems for set up, management and controlled experiments. In our ongoing work, we are building a software infrastructure that allows carrying out experiments with sensor networks of dimensions that are too large to connect each single sensor over a serial line with a PC.

2 Setting up a large WSN

In order to set up and maintain a large wireless sensor network, at least in the research and experimental phase, over-the-air software updates and patches are needed. The new, slightly updated version ESB/2 (prototype shown in Figure 1) of our sensor nodes allows us to perform updates of "user-space" tasks without the risk of flashing a node into a useless state. Besides a good design of the node enabling this, additional questions arise. For example, how to distribute updates across a network and how to ensure consistency of versions across the network.

Figure 1: Sensor Node

2.1 Over-the-Air Updates of Sensor Nodes

A main goal we followed when implementing over-the-air updates was to enable downloading new tasks onto a node while maintaining a stable core. Therefore, we separated the formerly monolithic software running on the nodes into a "firmware" core and tasks running on top of the firmware. The technical basis is a separation of the memory into a firmware area and an area where the tasks reside (cf. memory map given in Figure 2). The task code has to be linked against the firmware binary and thus can use all methods defined in it. In addition, tasks can register callback functions for special events like sensor data or packet types. If packets of a type that no task registered for are received, they can be immediately discarded. The same holds for sampling sensor values that can be skipped for those sensors no task registered for.

2.2 Software Distribution and Consistency

The software updates are first copied into the EEPROM that can store up to 64 kByte of code. In the second step the code is written into the flash. This two-step approach has advantages

over direct streaming of code from the air interface: It allows checking the integrity of the received code and it allows synchronizing updates across the network.

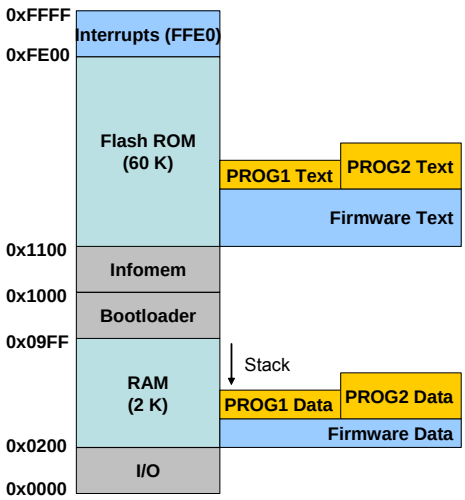

Figure 2: Memory Map

New versions could be distributed across a network using some known protocol, for example directed diffusion. Checksums and local retransmissions could be used to make sure the code was not altered on the path. At the end of this phase all (reachable) nodes have the new code image available in the EEPROM and are ready to write it to the flash ROM. In order to get a maximum of consistency flashing can be postponed until either a flash command is distributed across the network or a point of time in a time-synchronized network is reached. Depending on the amount of changes it will be crucial to synchronize the updates, e.g. if new routing protocols are deployed or sensor coverage must be guaranteed all the time.

3 Managing the state of the WSN

We started with a simple terminal application for issuing single commands to the network (left side of Figure 3). It allows the user to read sensor values, gradient and neighbour table and even to "log in" remotely at another node. Based on these simple commands we next implemented a GUI (both on Windows CE and Windows XP) that visualizes the sensor values and gives add-on features like an alarm if user-defined threshold values of some sensors are passed (cf. right side of Figure 3).

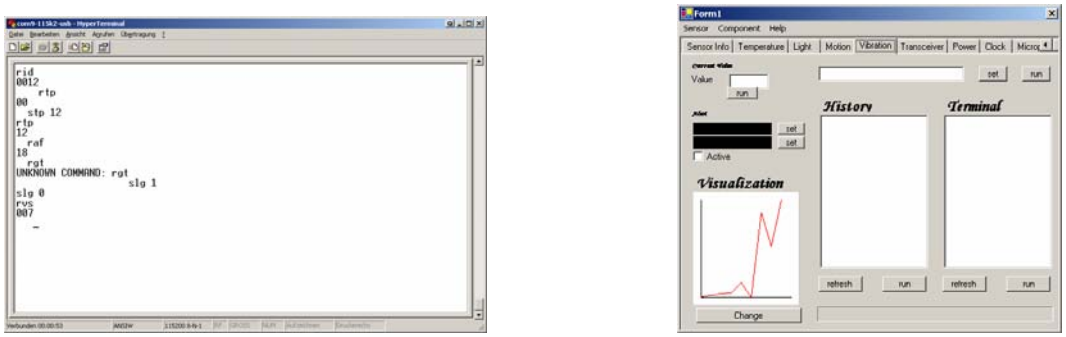

Figure 3: Terminal application and GUI

While these are still stand alone applications, integration of sensor networks into the Internet requires a larger effort. Our goal is to present the user highly aggregated data and to give him a high-level management interface running in a browser somewhere in the Internet. Therefore

we deploy an infrastructure consisting of three elements (cf. Figure 4): The sensor network is attached over a serial cable with a PC running an XP service that makes the serial port communication available for external clients. Client of the XP service is in our case a proxy with an http interface for browser connection. Data aggregation and filtering can be performed by modules attached to the proxy. The proxy finally provides Web access to a sensor network using a browser.

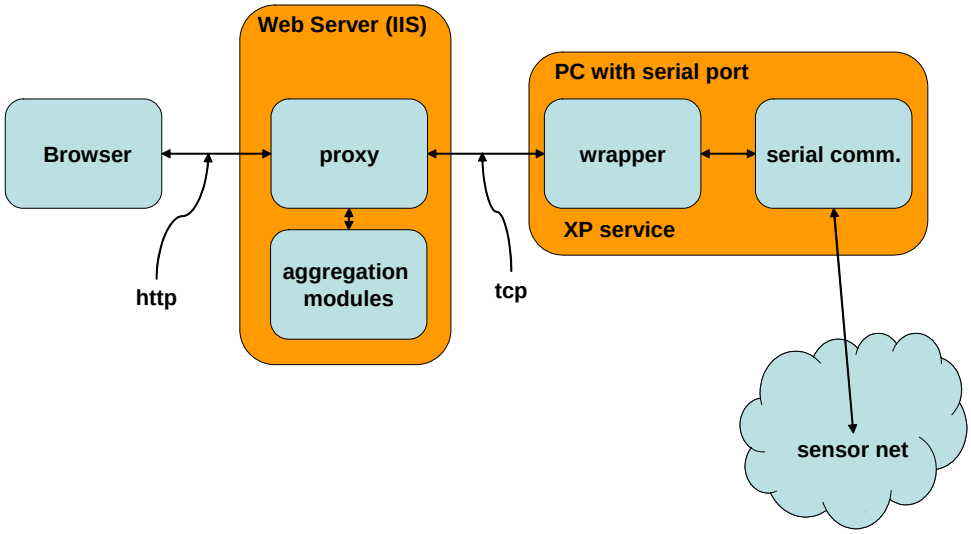

Figure 4: Remote access to a sensor network

This architecture allows a clear conceptual separation of the basic communication (right-hand side of the Figure 4), the presentation using the browser and stateful aggregation and development modules attached to the proxy at the Web Server.

4 Outlook: Performing reproducible experiments

We are currently developing different aggregation modules that do not only allow reading sensor data in a browser interface, but additionally enable authorized users to perform software updates and scripted experiments using large sensor networks. Consider for example experiments that compare a routing protocol in simulation and reality. Using a simple routing mechanism built into the firmware the new routing protocol could be loaded as a new task onto all nodes and activated. Each node records relevant data like amount of packets sent and received or energy consumption. A task could as well emulate pseudo-random or scripted sensor events. Scripting allows to control all parts of the experiment and to get reproducible results. After a given time, the task finishes and the data of all sensor nodes is collected, aggregated in an evaluation module and presented to the researcher.

5 Acknowledgements

Special thanks to our students Björn Lichtblau, Katrin Radloff and Tomasz Naumowicz who were involved in different parts of the software described in this paper.

Programming Paradigms and Middleware for Sensor Networks

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

Abstract

Programming sensor networks is currently a cumbersome and error-prone task since it requires programming individual sensor nodes, using low-level programming languages, interfacing to the hardware and the network, only supported by primitive operating system abstractions. There is a strong need for programming abstractions that simplify tasking sensor networks, and for middleware that supports such programming abstractions. We outline challenges in the design of such abstractions and middleware. Also, we present and discuss currently examined approaches to sensor network programmability.

1 Motivation

Distributed programming abstractions like Remote Procedure Calls (RPC), the Distributed Object Model (DOM), or Distributed Shared Memory (DSM) have traditionally simplified and enabled the implementation of complex distributed systems. These programming abstractions served as foundations for successful middleware architectures such as ONC RPC [13], CORBA [15], or MUNIN [5]. Unfortunately, these abstractions and middleware architectures cannot be simply applied to sensor networks due to the new characteristics and peculiarities of the latter. Existing approaches have to be revisited or new approaches have to be developed to meet the requirement of sensor networks [11, 14, 16]. We will sketch these requirements below.

Programming Paradigm. Sensor networks are used to monitor a wide variety of environmental phenomena. Hence, mechanisms are needed for the specification of high-level sensing tasks, for automated programming of individual nodes according to a sensing task, and for combining sensory data from individual sensor nodes into high-level sensing results. A programming paradigm should substantially support the development of applications according to this model. In particular, it should hide hardware and distribution issues from the programmer as far as possible. Ideally, a programming paradigm allows to program the sensor network as a single "virtual" entity, rather than focusing on individual nodes. While providing easy to use abstractions on the one hand, the paradigm should allow the efficient implementation of a wide variety of different applications. Typically, there are tradeoffs between the level of abstraction, expressiveness, and the efficiency of implementations.

Restricted Resources. Sensor nodes suffer from limited resources like energy, computing power, memory, and communication bandwidth. Hence, middleware components have to be lightweight to fit these constraints. Middleware should also provide support to dynamically adapt performance and resource consumption to the varying needs of the application, for example, by enabling dynamic tradeoffs between output quality and resource consumption. Since it is anticipated that a sensor network will execute multiple applications concurrently, middleware should also provide mechanisms to optimize resource allocation for overall system performance.

Network Dynamics. Ad hoc networks of sensor nodes may exhibit a highly dynamic network topology due to node mobility, environmental obstructions resulting in communication failures (e.g., a truck driving by), or hardware failures (e.g., depleted batteries, stepping on a device). Middleware should support the robust operation of

sensor networks despite these dynamics by adapting to the changing network environment. One prominent approach is data-centric communication, where network nodes are no longer addressed by unique identifiers, but by specifying the function or data they provide (e.g., "some device in my vicinity that can measure temperature"), allowing for automatic failover to a device with characteristics similar to the failed one.

Scale of Deployments. As sensor nodes become smaller and cheaper, sensor networks are anticipated to consist of thousands or millions of individual nodes. This precludes manual configuration, maintenance, fixing, or upgrading of individual devices due to their huge number. Hence, middleware should support mechanisms for self-configuration and self-maintenance of collections of sensor nodes. In an extreme case, starting from a totally symmetric situation (all devices are identical initially and do not possess unique identifiers), the collection of devices must self-configure in order to achieve an operational state (e.g., set up a network topology, assign tasks to devices, collaboratively merge and evaluate collected data). Although such mechanisms try to achieve a certain global structure or state of the network, it is often impossible to apply global approaches (e.g., flooding the network) to achieve these due to scalability reasons. Fortunately, it is often possible to approximate a global goal by using localized mechanisms, where devices interact only with their close neighborhood.

Real-world Integration. Sensor networks are used to monitor real-world phenomena. Hence, the categories time and space play a crucial role in sensor networks to identify events in the real world (i.e., time and place of event occurrence), to distinguish events in the real world (i.e., separation of events in time and space), and to correlate information from multiple sources. Hence, the establishment of a common time scale and location grid among sensor nodes is an important middleware service. Additionally, many applications are subject to real-time constraints, for example, to have the sensor network report a certain state of the real world within a certain amount of time. Hence, middleware may have to include real-time mechanisms.

Collection and Processing of Sensory Data. The collection and processing of sensory data is a core functionality in sensor networks. Complex sensing tasks often require that data collected at many nodes be fused to obtain a high-level sensing result. However, limited resources preclude centralized data processing, since that requires transferring bulky raw sensor data from many nodes through the network. Instead, sensory data is preprocessed at the source to extract features relevant to the application. Such preprocessed data from multiple sources is fused and aggregated in the network as it flows towards the "user", further reducing communication and energy overhead. These techniques to some degree blur the clear separation of communication and data processing typically found in traditional distributed systems. Middleware with support for the above data reduction techniques will require means to specify application knowledge and ways to inject this knowledge into the nodes of the network.

Integration with Background Infrastructures. Sensor networks are rarely stand-alone systems. Rather, they are connected to external devices and infrastructures for several reasons. Firstly, this may be necessary for tasking the sensor network, as well as for evaluation and storage of sensing results. Secondly, background infrastructures such as the Internet may provide resources (e.g., computing power, storage) which are not available in the sensor network. Hence, middleware should allow the integration with such background infrastructures, providing a homogeneous view on a wide variety of extraordinarily heterogeneous computing devices. Additionally, sensor nodes might not possess the resources to actually execute specific middleware components. In such cases, it might be a viable option to source out middleware components to the background infrastructure, only keeping minimal functionality on the devices. Note that this minimal set of on-device functionality might change over time in order to adapt to a changing environment, thus necessitating mechanisms for dynamically updating services executing on the devices.

Interfacing with the Operating System. Traditional middleware is typically designed to sit on top of established operating systems, which already provide rich abstractions such as task and memory management. In sensor networks, however, operating system abstractions and concrete operating systems for sensor nodes are currently an area of active research. Hence, the functional sep-

aration and the interface between operating system and middleware is not well understood. Due to resource constraints, it is likely that operating system functionality (e.g., task and memory management) will be rather primitive compared to traditional OS [6]. One possible option is to give up the separation between operating system and middleware and aim for a distributed operating system that unifies traditional OS and middleware functionality.

2 Middleware Approaches

Databases. A number of approaches [3, 9, 12] have been devised that treat the sensor network as a distributed database where users can issue SQL-like queries to have the network perform a certain sensing task. We will discuss TinyDB [9] as a representative of this class.

TinyDB supports a single "virtual" database table `sensors`, where each column corresponds to a specific type of sensor (e.g., temperature, light) or other source of input data (e.g., sensor node identifier, remaining battery power). Reading out the sensors at a node can be regarded as appending a new row to `sensors`. The query language is a subset of SQL with some extensions. Consider the following query example. Several rooms are equipped with multiple sensor nodes each. Each sensor node is equipped with sensors to measure the acoustic volume. The table `sensors` contains three columns `room` (i.e., the room number the sensor is in), `floor` (i.e., the floor on which the room is located), and `volume`. We can determine rooms on the 6th floor where the average volume exceeds the threshold 10 with the following query:

```
SELECT AVG(volume), room FROM sensors
  WHERE floor = 6
  GROUP BY room
  HAVING AVG(volume) > 10
  EPOCH DURATION 30s
```

The query first selects rows from sensors at the 6th floor (`WHERE floor = 6`). The selected rows are grouped by the room number (`GROUP BY room`). Then, the average volume of each of the resulting groups is calculated (`AVG(volume)`). Only groups with an average volume above 10 (`HAVING AVG(volume) > 10`) are kept. For each of the remaining groups, a pair of average volume and the respective room number (`SELECT AVG(volume), room`) is returned. The query is re-executed every 30 seconds (`EPOCH DURATION 30s`), resulting in a stream of query results.

TinyDB uses a decentralized approach, where each sensor node has its own query processor that preprocesses and aggregates sensor data on its way from the sensor node to the user. Executing a query involves the following steps: Firstly, a spanning tree of the network rooted at the user device is constructed and maintained as the network topology changes, using a controlled flooding approach. The flood messages are also used to roughly synchronize time among the nodes of the network. Secondly, a query is broadcast to all the nodes in the network by sending it along the tree from the root towards the leafs. During this process, a time schedule is established, such that a parent and its children agree on a time interval when the parent will listen for data from its children. At the begin of every epoch, the leaf nodes obtain a new table row by reading out their local sensors. Then, they apply the select criteria to this row. If the criteria are fulfilled, a so-called partial state record is created that contains all the necessary data (i.e., room number, floor number, average volume in the example). The partial state record is then sent to the parent during the scheduled time interval. The parent listens for any partial state records from its children during the scheduled interval. Then, the parent proceeds as the children by reading out its sensors, applying select criteria, and generating a partial state record if need be. Then, the parent aggregates its partial state record and the records received from its children (i.e., calculates the average volume in the example), resulting in a new partial state record. The new partial state record is then sent to the parent's parent during the scheduled interval. This process iterates up to the root of the tree. At the root, the final partial state record is evaluated to obtain the query result. The whole procedure repeats every epoch.

Mobile Agents. Another class of middleware approaches is inspired by mobile code and mobile agents [4, 7]. There, the sensor network is tasked by injecting a program into the sensor network. This program can collect local sensor data, can statefully migrate or copy itself to other nodes, and can communicate with such remote copies. We discuss SensorWare [4] as a representative of this class.

In SensorWare, programs are specified in Tcl [10], a dynamically typed, procedural programming language.

The functionality specific to SensorWare is implemented as a set of additional procedures in the Tcl interpreter. The most notable extensions are the `query`, `send`, `wait`, and `replicate` commands. `query` takes a sensor name (e.g., `volume`) and a command as parameters. One common command is `value` which is used to obtain a sensor reading. `send` takes a node address and a message as parameters and sends the message to the specified sensor node. Node addresses currently consist of a unique node ID, a script name, and additional identifiers to distinguish copies of the same script. The `replicate` command takes one or more sensor node addresses as parameters and spawns copies of the executing script on the specified remote sensor nodes. Node addresses are either unique node identifiers or "broadcast" (i.e., all nodes in transmission range). The `replicate` command first checks whether a remote sensor node is already executing the specified script. In this case, there are options to instruct the runtime system to do nothing, to let the existing remote script handle this additional "user", or to create another copy of the script. In SensorWare, the occurrence of an asynchronous activity (e.g., reception of a message, expiry of a timer) is represented by a specific event each. The `wait` command expects a set of such event names as parameters and suspends the execution of the script until one of the specified events occurs.

The following script is a simplified version of the TinyDB query and calculates the maximum volume over all rooms (i.e., over all sensor nodes in the network):

```
set children [replicate]
set num_children [llength $children]
set num_replies 0
set maxvolume [query volume value]
while {1} {
  wait anyRadioPck
  if {$maxvolume < $msg_body} {
    set maxvolume $msg_body }
  incr num_replies
  if {$num_replies = $num_children} {
    send $parent $maxvolume
    exit }
}
```

The script first replicates itself to all nodes in communication range. No copies are created on nodes already running the script. The `replicate` command returns a list of newly "infected" sensor nodes (`children`). Then, the number of new children (`num_children`) is calculated, the reply counter (`num_replies`) is initialized to zero, and the volume at this node is measured (`maxvolume`). In the loop, the `wait` blocks until a radio message is received. The message body is stored in the variable `msg_body`. Then, `maxvolume` is updated according to the received value and the reply counter is incremented by one. If we received a reply from every child, then `maxvolume` is sent to the parent script and the script exits. Due to the recursive replication of the script to all nodes in the network, the user will eventually end up with a message containing the maximum volume among all nodes of the network.

Events. Yet another approach to sensor network middleware is based on the notion of events. There, the application specifies interest in certain state changes of the real world ("basic events"). Upon detecting such an event, a sensor node sends a so-called event notification towards interested applications. The application can also specify certain patterns of events ("compound events"), such that the application is only notified if occurred events match this pattern. We discuss DSWare [8] as a representative of this class.

DSWare supports the specification and automated detection of compound events. A compound event specification contains, among others, an event identifier, a detection range specifying the geographical area of interest, a detection duration specifying the time frame of interest, a set of sensor nodes interested in this compound event, a time window W, a confidence function f, a minimum confidence $c_{\min}$, and a set of basic events E. The confidence function f maps E to a scalar value. The compound event is detected and delivered to the interested sensor nodes, if $f(E) \geq c_{\min}$ and all basic events occurred within time window W.

Consider the example of detecting an explosion event, which requires the occurrence of a light event (i.e., a light flash), a temperature event (i.e., high ambient temperature), and a sound event (i.e., a bang sound) within a subsecond time window W. The confidence function is defined as:

$$f = 0.6 \cdot B(\text{temp}) + 0.3 \cdot B(\text{light}) + 0.3 \cdot B(\text{sound})$$

The function B maps an event ID to 1 if the respective

event has been detected within the time window W, and to 0 otherwise. With $c_{\min} = 0.9$, the above confidence function would trigger the explosion event if the temperature event is detected along with one or both of the light and sound events. This confidence function expresses the fact that detection of the temperature event gives us higher confidence in an actual explosion happening than the detection of the light and sound events.

Additionally, the system includes various real-time aspects, such as deadlines for reporting events, and event validity intervals.

Other Approaches. Besides the ones discussed above, there are yet other approaches to programming sensor networks. MagnetOS [2], for example, is a distributed virtual machine, where an object oriented byte-code program is automatically partitioned and allocated to the nodes of a sensor network, such that communication overhead among the distributed components is minimized. EnviroTrack [1] is a middleware specifically tailored to observing mobile physical entities in the environment. As the observed entity moves through the sensor field, nearby sensor nodes form temporary groups to observe the target. Such a dynamically changing group is exposed as an addressable entity to the programmer. The programmer can attach so-called tracked objects to this entity, containing arbitrary data, computation, or actuation. The tracked object is executed by the (dynamically changing) sensor group.

3 Discussion

In the previous section, we presented concrete middleware approaches for sensor networks with different underlying programming paradigms (e.g., database approach, agent-based approach, event-based approach). These paradigms are not new, but required significant adaptation for use in sensor networks. The approaches differ with respect to ease of use, expressiveness, scalability, and overhead. We will discuss the above middleware approaches with respect to these criteria.

TinyDB provides the user with a declarative query system which is very easy to use. The database approach hides distribution issues from the user. Rather than programming individual sensor nodes, the network of nodes is programmed as a single (virtual) entity. The user is relieved from interfacing with low-level APIs on the sensor node. On the other hand, the expressiveness of the database approach is limited in various ways. Firstly, adding new aggregation operations is a complex tasks and requires modifications of the query processor on all sensor nodes. But more importantly, it is questionable whether more complex sensing tasks can be appropriately supported by a database approach. For example, the system does not explicitly support the detection of spatio-temporal relationships among events in the real-world (e.g., expressing interest in a certain sequence of events in certain regions). The system also suffers from some scalability issues, since it establishes and maintains network-wide structures (e.g., spanning tree of the network, queries are sent to all nodes). In contrast, many sensing tasks exhibit very local behavior (e.g., tracking a mobile target), where only very few nodes are actively involved at any point in time. This suggests that TinyDB cannot provide optimal performance for such queries. Additionally, the tree topology used by TinyDB is independent of the actual sensing task. It might be more efficient to use application-specific topologies instead.

SensorWare uses an imperative programming language to task individual nodes. Even rather simple sensing tasks result in complex scripts that have to interface with operating system functionality (e.g., querying sensors) and the network (e.g., sending, receiving, and parsing messages). On the other hand, SensorWare's programming paradigm allows the implementation of almost arbitrary distributed algorithms. Typically, there is no need to change the runtime environment in order to implement particular sensing tasks. However, the low performance of interpreted scripting languages might necessitate the native implementation of complex signal processing functions (e.g., Fast Fourier Transforms, complex filters), thus requiring changes of the runtime environment in some cases. SensorWare allows the implementation of highly scalable applications, since the collaboration structures among sensor nodes are up to the application programmer. For example, it is possible to implement activity zones of locally cooperating groups of sensor nodes that "follow" a tracked target. The SensorWare runtime does not maintain any global network structures. One potential problem is the address-centric nature of SensorWare, where specific nodes are addressed by unique identifiers. This may

lead to robustness issues in highly dynamic environments as mentioned in Section 1.

DSWare provides compound events as a basic programming abstraction. However, a complete sensor network application will require a number of additional components besides compound event detection. For example, code is needed to generate basic events from sensor readings, or to act on a detected compound event. DSWare does not provide support for this glue code, requiring the user to write low-level code that runs directly on top of the sensor node operating system. This makes the development of any application a complex task, while at the same time providing a maximum of flexibility. DSWare supports only a very basic form of compound events: the logical "and" of event occurrences enhanced by a confidence function. It might be worthwhile to consider more complex compound events, such as explicit support for spatio-temporal relationships among events (e.g., sequences of events, non-occurrence of certain events). Note that more restrictive compound event specifications can avoid the transmission of event notifications and can hence contribute to better energy efficiency and scalability.

Overall, the examined middleware approaches exhibit a tradeoff between ease of use and expressiveness. While TinyDB is easy to use, it is restricted to a few predefined aggregation functions. More complex queries either require changes in the runtime environment, are inefficient, or cannot be expressed at all. While SensorWare and DSWare support the efficient implementation of almost arbitrary queries, even simple sensing tasks require significant programming efforts. Narrowing this gap between ease of use and expressiveness while concurrently enabling scalable and energy-efficient applications is one of the major challenges in the design of sensor network middleware. It is not yet clear, whether suitable programming abstractions will be inspired by known paradigms as in the three presented examples. Possibly, completely new approaches have to be developed to meet the above goals.

References

[1] T. Abdelzaher, B. Blum, Q. Cao, Y. Chen, D. Evans, J. George, S. George, L. Gu, T. He, S. Krishnamurthy, L. Luo, S. Son, J. A. Stankovic, R. Stoleru, and A. Wood. EnviroTrack: Towards an Environmental Computing Paradigm for Distributed Sensor Networks. In *ICDCS 2004*, Tokyo, Japan, March 2004.

[2] R. Barr, J. C. Bicket, D. S. Dantas, B. Du, T. W. D. Kim, B. Zhou, and E. G. Sirer. On the Need for System-Level Support for Ad Hoc and Sensor Networks. *Operating System Review*, 36(2):1–5, 2002.

[3] P. Bonnet, J. E. Gehrke, and P. Seshadri. Querying the Physical World. *IEEE Personal Communications*, 7(5):10–15, 2000.

[4] A. Boulis, C.C. Han, and M. B. Srivastava. Design and Implementation of a Framework for Programmable and Efficient Sensor Networks. In *MobiSys 2003*, San Franscisco, USA, May 2003.

[5] J. B. Carter, J. K. Bennet, and W. Zwaenepoel. Implementation and Performance of Munin. In *SOSP 1991*, Pacific Grove, USA, October 1991.

[6] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System Architecture Directions for Networked Sensors. In *ASPLOS 2000*, Cambridge, USA, Nov. 2000.

[7] P. Levis and D. Culler. Mate: A Tiny Virtual Machine for Sensor Networks. In *ASPLOS X*, San Jose, USA, October 2002.

[8] S. Li, S. H. Son, and J. A. Stankovic. Event Detection Services Using Data Service Middleware in Distributed Sensor Networks. In *IPSN 2003*, Palo Alto, USA, April 2003.

[9] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. TAG: a Tiny Aggregation Service for Ad-Hoc Sensor Networks. In *OSDI 2002*, Boston, USA, December 2002.

[10] J. K. Ousterhout. *Tcl and the Tk Toolkit*. Addison-Wesley, 1994.

[11] K. Römer, O. Kasten, and F. Mattern. Middleware Challenges for Wireless Sensor Networks. *ACM SIGMOBILE Mobile Computing and Communication Review (MC2R)*, 6(4):59–61, October 2002.

[12] C. C. Shen, C. Srisathapornphat, and C. Jaikaeo. Sensor Information Networking Architecture and Applications. *IEEE Personal Communications*, 8(4):52–59, 2001.

[13] R. Srinivasan. RPC: Remote Procedure Call Protocol Specification Version 2 (RFC 1831), 1995.

[14] J. A. Stankovic, T. Abdelzaher, C. Lu, L. Sha, and J. Hou. Real-Time Communication and Coordination in Embedded Sensor Networks. *Proceedings of the IEEE*, 91(7), 2003.

[15] S. Vinoski. CORBA: Integrating Diverse Applications Within Distributed Heterogeneous Environments. *IEEE Communications Magazine*, February 1997.

[16] Y. Yu, B. Krishnamachari, and V. K. Prasanna. Issues in Designing Middleware for Wireless Sensor Networks. *IEEE Network Magazine*, 2003. To appear.

Content Evolution Driven Data Propagation in Wireless Sensor Networks

Gero Mühl, Andreas Ulbrich
TU Berlin, iVS, Einsteinufer 17, 10587 Berlin, Germany
{gmuehl,ulbi}@ivs.tu-berlin.de

Hartmut Ritter
Freie Universität Berlin, Takustr. 9, 14195 Berlin
hritter@inf.fu-berlin.de

Abstract

Wireless sensor networks are currently a viable field of active research. In sensor networks, communication often resembles the publish/subscribe paradigm where source nodes publish data which is subsequently delivered to a set of subscribed sink nodes. We propose to extend routing algorithms known from publish/subscribe systems with data propagation based on the evaluation of the sensed values over time to make them suitable for sensor networks.

1 Introduction

A *sensor network* consists of a large amount of small *sensor nodes* built of inexpensive hardware with limited computing, memory, networking, and battery power. Usually, each sensor node is equipped with some *sensors* (e.g., for temperature, pressure, noise). Since, the sensor values are snooped from the physical world, they continously change over time. The sensor nodes communicate with each other in an ad hoc fashion: Each sensor can communicate only with a set of neighbor nodes which are in transmission range making communication possible. In general, the larger the distance to a neighbor, the more energy is necessary for transmission. Often a direct transmission between nodes is possible, but a transmission in several steps using intermediate nodes preserves energy. Unsurprisingly, *energy-preserving message routing* is one of the main issues in sensor networks.

It seems to be common sense that *synchronous request/reply-based* communication is not sensible for sensor networks. Hence, communication often resembles the known *publish/subscribe* paradigm where sensor nodes *publish* sensed data which is subsequently delivered to *subscribed* nodes that evaluate or aggregate the delivered data. For example, a consumer might subscribe to all sensed temperature values within a given geographical region. In publish/subscribe systems, routing algorithms that built upon routing tables held by each individual node are used to avoid flooding every sensed data into the whole network to reach their recipients. This is in line with the simple routing algorithm used by *directed diffusion* [1]. Routing algorithms known from publish/subscribe systems, however, so far have concentrated on acyclic topologies and discrete events mismatching the fault-tolerance requirements of sensor networks and the continuous nature of the sensed values. In the following we sketch, how these routing algorithms can be made feasible for sensor networks by extending them with data propagation based on the evolution of the sensed values over time.

2 Data Propagation using Funnel Functions

Efficient interest and data propagation is perhaps one of the most crucial tasks in sensor networks. In many cases, a fixed *rate-based* data distribution that periodically (e.g., every $10ms$) disseminates the sensed values is inappropriate because it might swamp consumers and the network with notifications not carrying sufficiently diverging information. An unnecessarily high message volume is generated by this rather simplistic and purely *time dependent* data distribution wasting battery power and network bandwidth. One might think of a *content dependent* distribution as a better solution. Such a dependency may either refer

to the *absolute value* or to the *relative change* with respect to the last value forwarded. The former category is often denoted with *content-based filtering* because a consumer expresses his interests using predicates over the content of notifications. For example, a consumer might only be interested in the temperature if it is above 20° and if the sensor is within a given geographical area. Filters are usually expressed using a certain *filter language*. Combinations of rate-based distribution and content-based filtering are currently supported by a number of systems (e.g. [1]). For example, if notifications consist of name/value pairs (e.g., $\{(temperature, 25), (location, [125, 220]\})$, an example for a subscription may be $temperature > 20 \wedge location\ in\ [-100, 200, 300, 500]) \wedge rate = 50ms$. The main disadvantage of this strategy is that the difference among consecutive temperature notifications may be so small that they are relevant for the application. An example, for the latter category referring to a relative change is a consumer that is only interested in a sensed temperature if it diverges by at least 1° from the value previously delivered. However, this strategy, has the disadvantage that a consumer cannot easily distinguish whether a sensed value has not changed sufficiently to lead to a new notification or whether new data is not delivered due to some system failure. Moreover, in many application scenarios a change in the sensed values is more interesting if the change occurred in a small time interval. For example, a change in the temperature of about 10° in about 1 minute is clearly more interesting than the same change from morning to noon. What is really needed in our view is a flexible data propagation based on the sensed values *and* their evolution over time.

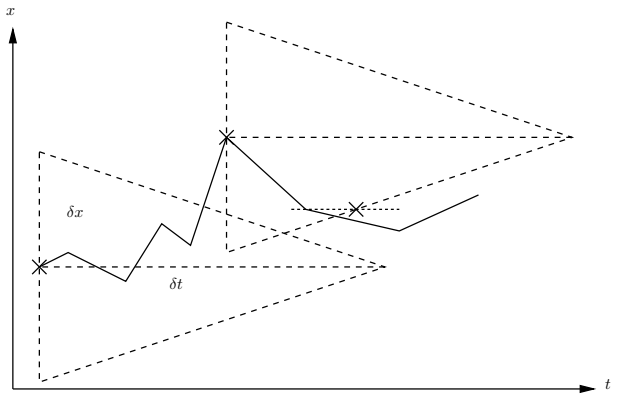

Figure 1. Triangularly shaped linear dependency among δx and δt.

In our model, the consumers attach a description of their specific and application dependent data needs to their subscription, extending its expressiveness. This strategy not only allows a consumer to avoid subscribing to irrelevant data sequences, but perhaps more importantly, it also enables the network to optimize data distribution saving network bandwidth and battery power. For example, assume that a temperature x was delivered to a consumer and that the change of the temperature and the elapsed time is given by δx and δt, respectively. In this case, a consumer might specify that it is only interested in a new notification if $|\delta x| > 1° \cdot (1 - \delta t/10s)$. This simple triangularly shaped linear dependency (see Fig. 1) delivers the next sensor value immediately if it diverges by at least 1°. If the change is less, the delivery is delayed until it is sufficient or the next snooped sensor value becomes available for evaluation, whatever comes first. In any case, the most current sensor value is sent out after 10 seconds, serving as a heart beat. We call such a function that describes the necessary relative change to trigger the forwarding of a sensor value over time a *funnel function*. Triangularly shaped funnel functions have the general form $|\delta x| > c \cdot (1 - \delta t/d)$. Different categories of funnel functions can be thought of. For example, linear, step-wise linear, inverse linear, or even application dependent functions. However, in our view, only funnel functions that monotonically decrease with the time are sensible.

Currently, we are investigating how to extend routing algorithms to incorporate funnel functions and to work with cyclic topologies. The more advanced routing algorithms exploit similarities among the subscriptions (identity and coverage tests as well as filter merging [2]) to reduce the routing table sizes and the amount of messages exchanged for their actualization. For example with covering-based routing, a filter does not need to be propagated to a neighbor node if a covering filter matching a superset of notifications has been propagated to that neighbor before. Additionally, if a node receives a filter from a neighbor node it can remove all those routing entries regarding this neighbor from its routing table whose filter is covered by the new filter. Merging-based routing generates from given a set of filters a new filter that covers the original filters. This broader filter is then propagated instead of its constituting filters. We say that the merger is *perfect* if it matches only notifications that one of its constituting filter matches, otherwise, the merger is *imperfect*. In general, the use of the more advanced routing algorithms have some requirements: Similar to the expressiveness of filter predicates, the funnel function has to be limited to certain types. This restriction is necessary because these algorithms require that iden-

tity tests, coverage tests, and merging among filters including the funnel function can be efficiently computed. One main goal is to determine useful funnel functions under this condition.

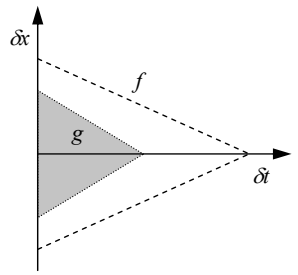

Figure 2. Funnel function g covers f.

A funnel function covers another funnel function if the hull of the former function lies completely within the hull of the latter function (see Fig. 2). This means that a triangularly shaped funnel function $|\delta x| > c_1 \cdot (1 - \delta t/d_1)$ covers another triangularly shaped funnel function $|\delta x| > c_2 \cdot (1 - \delta t/d_2)$ if $c_1 \leq c_2$ and $d_1 \leq d_2$. For example, the funnel function $|\delta x| > 1° \cdot (1 - \delta t/10s)$ covers the funnel function $|\delta x| > 2° \cdot (1 - \delta t/20s)$.

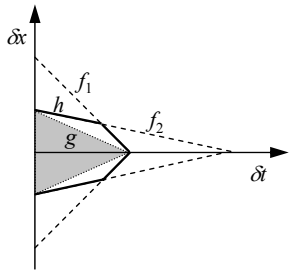

Figure 3. Merging of two funnel functions: h is a perfect merging of f_1 and f_2, g is an imperfect merging.

Funnel functions that do not cover each other can be merged to a new funnel function that covers all funnel functions it was generated from. The hull of the merged funnel function is given by the intersection of the hulls of its constituting funnel functions. Perfect merging of triangularly shaped funnel functions leads to step-wise linear functions that are complete under merging (see Fig. 3). If the merger should again be a triangularly shaped funnel function, the smallest triangle that lies within the hull of all merged triangularly shaped funnel functions can be used as imperfect merger.

Regarding the use of cyclic topologies, we are investigating two scenarios. The first scenario floods subscriptions into the network similar to directed diffusion. In the second scenario, the advanced routing algorithms are run on top of a self-stabilizing spanning tree algorithm. The spanning tree algorithm runs independently from the routing algorithms. It maintains a spanning tree of the sensor-network. The spanning tree is then used by the publish/subscribe routing algorithms for notification delivery.

3 Conclusions

In this paper, we have presented first ideas to reuse content-based routing algorithms known from publish/subscribe systems for sensor networks. The extensions to exploit the evolution of the sensed values over time in the data distribution seems to be a good starting point for further investigations and experiments. We are planning to test the approach on own hardware. Here, a main challenge will be to support filtering in a stage of sensor data processing as early as possible in order to limit the power consumption.

We are building a publish/subscribe middleware called REBECA including a version that runs on small sensor nodes. Programs for the REBECA platform are designed by exploiting a model driven architecture (MDA) approach. The programs are visually designed using an existing tool in the platform independent model (PIM) domain. Then, the code for the platform specific model (PSM), i.e., the sensor hardware, is generated automatically by appropriate code transformations [3]. This allows to abstract as much a possible from the underlying implementation. The model transformation will take care of the limitations of sensor nodes to a large extent. The generated code is then downloaded to the sensor nodes.

References

[1] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva. Directed diffusion for wireless sensor networking. *IEEE/ACM Transactions on Networking (TON)*, 11(1):2–16, 2003.

[2] G. Mühl. *Large-Scale Content-Based Publish/Subscribe Systems*. PhD thesis, Darmstadt University of Technology, 2002. http://elib.tu-darmstadt.de/diss/000274/.

[3] T. Weis, A. Ulbrich, and K. Geihs. Model metamorphosis. *IEEE Software*, 20(5):46–51, September/October 2003.