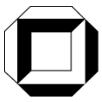
Universität Karlsruhe - Fakultät für Informatik - Bibliothek - Postfach 6980 - 76128 Karlsruhe

Mobile und Verteilte Systeme
Ubiquitous Computing
Teil IV

Herausgeber:
Till Riedel, Christian Decker, Patrik Spieß,
Luciana Moreira Sá de Souza

Interner Bericht 2007-3



Universität Karlsruhe (TH)
Forschungsuniversität · gegründet 1825

ISSN 1432-7864

Fakultät für **Informatik**

Vorwort

Das Seminar „Mobile und verteilte Systeme – Ubiquitous Computing“ am Telecooperation Office (TecO) erfreut sich großer Beliebtheit. Der vorliegende interne Bericht enthält die studentischen Beiträge zu diesem Seminar, das im WS 2006/07 in dieser Form zum fünften Mal stattgefunden hat.

Die Themenauswahl für das Seminar orientiert sich im wesentlichen an aktuellen wissenschaftlichen Fragestellungen in den Bereichen:

- Kontexterfassung und Verarbeitung
- Kommunikation in ubiquitären Informationssystemen
- Sicherheitsaspekte in ubiquitären Systemen
- Technologien für das Ubiquitous Computing
- Auswirkungen ubiquitärer Technologien
- Softwaretechnik für ubiquitären Informationssystemen

Aufgrund des stetig wachsenden Interesses der Studenten an diesen Themenbereichen haben wir uns entschlossen einen Seminarband mit den Beiträgen unserer Studenten als internen Bericht zu veröffentlichen. Durch die engagierte Mitarbeit der beteiligten Studenten wird ein Ausschnitt aus diesem komplexen und umfassenden Themengebiet klar und übersichtlich präsentiert. Für den Fleiß und das Engagement unserer Seminarteilnehmer wollen wir uns an dieser Stelle daher herzlich bedanken.

Bestärkt durch die gute Resonanz der Studenten, werden wir dieses Seminar auch im nächsten Wintersemester – dann natürlich mit aktualisierten Themen – wieder anbieten.

Karlsruhe, März 2007

Till Riedel, Christian Decker, Patrik Spieß, Luciana Moreira Sá de Souza

Inhalt

Internet der Dinge
Telekommunikationsinfrastruktur

Daniel Kühner

Telecooperation Office, Institut für Telematik, Universität Karlsruhe (TH)
Betreuer Till Riedel

3. Februar 2007

1 Einleitung

Internet der Dinge. Dieser Begriff verkörpert den Nachfolger des Web 2.0, wenn man den Zukunftsvisionen traut. Es beschreibt ein weltumspannendes Netzwerk von Geräten/Dingen und Nutzern, die miteinander verbunden sind, Zugriff aufeinander haben und zusammen kommunizieren können. Die Größe des Netzwerkes lässt sich nicht genau festlegen; es werden aber Milliarden oder Billionen von Geräten sein.

Dieses neuartige Konzept stellt einen zukunftsweisenden Plan für mobile und drahtlose Systeme auf. Es zeigt also, wie die heutigen Netzwerktechnologien weiterentwickelt werden können, wie wir in Zukunft mit Informationen umgehen werden und wie Kommunikation sein wird. Man muss dabei erwähnen, dass die einfachsten Formen davon schon heute Realität sind. Im Jahr 2005 gab es bereits 1,3 Milliarden RFID Tags und zwei Milliarden Nutzer mobiler Dienste. Der Umsatz lag 2005 bei weltweit 600 Millionen verkauften RFID-Tags und „für das Jahr 2006 erwartet man einen weltweiten Absatz von 1,3 Milliarden RFID-Tags“ [Wik07a].
Die Idee des Internets der Dinge entstand aus den fortgeschrittenen Konzepten der letzten 20 Jahre [Buc06]:

- Ubiquitous Communications - allgegenwärtige Kommunikation
- Pervasive Computing - Rechnerdurchdringung
- Ambient Intelligence - Umgebungsintelligenz

Schaut man sich das ganze Spektrum an, so muss man feststellen, dass das immer größer werdende Wachstum von verbundenen intelligenten Objekten nicht mehr aufzuhalten ist. Die Vernetzung der neuen System aus Dingen und Geräten und der wirtschaftliche Wert sind dabei nicht mehr zu trennen, denn der wirkliche Mehrwert ist die Vernetzung selbst. Der Technologieschub, der mit dem Internet der Dinge kommen wird, wird es uns ermöglichen diese neuartige Technologie unsichtbar und unbemerkt in unsere soziale Umgebung einzupflanzen. Sie wird unsere Wirtschaft und Gesundheit verbessern, unser Zusammenleben und Privatleben unterstützen.

Doch die Anwendungsgebiete scheinen endlos zu sein. Als Beispiele seien genannt [Buc06]:

- *Güter- und Warenmanagement.* Elektronische Überwachung und Remote Sensing bestimmen die Position von Gepäck auf einem Flughafen oder von Gütern in einem Produktionsprozess; im Supply Chain konnen Inhalte von einzelnen Verpackungen bis hin zu ganzen Containern innerhalb kleinster Zeiten erfasst werden.
- *Gesundheitswesen.* Blutdruck- und Herzrhytmussensoren übertragen regelmäßig Daten von zu Hause aus zu einem überwachungszentrum. Ein Computer entdeckt anormale Werte und informiert einen Arzt, der sich die Patientendaten betrachtet; Implantate beinhalten Blutgruppe und Unverträglichkeiten, die bei einem Unfall lebensrettend sein können.
- *Umgebungsüberwachung.* Ein Sensornetzwerk überwacht Flusspegel und Niederschlag, sagt Fluten vorher und unterstützt den Hochwasserschutz zur Flutbekämpfung; in Privathäuser und -anlagen überwachen Sensoren die Sicherheit und verständigen entsprechende Notdienste. Im Militärbereich denkt man schon an „Smart Dust“, ein Netzwerk aus staubkorngroßen Sensoren, die hinter feindlichen Linien die Umgebung ausspionieren und miteinander kommunizieren.

Nun liegt es an der Telekommunkationsbranche den Zugriff auf weiterführende Informationen zu ermöglichen, um die gespeicherten Daten in RFID Tags zu verarbeiten, entweder über das Internet oder das Mobilfunknetz. Immer mehr Menschen haben als ständigen Begleiter mobile Endgeräte dabei. Besonders in diesem Bereich ist mit einem neuen Technologieschub zu rechnen, da neue Dienste und Hardware die Funktionalität heutiger Endgeräte weit übersteigen wird. Denn mobile Endgeräte, die GPRS, GSM oder UMTS fähig sind, in Kombination mit einem passiven RFID-Tag, eröffnen eine große Bandbreite von verschiedenen Anwendungen im Bereich der mobilen Dienste und dem sogenannten „Physical Browsing“ [Sch04a]. Doch in diesem Zuge müssen die massiven neuen Datenaufkommen mit angepassten oder sogar neuen Netzwerken, wie zum Beispiel dem Mesh-Netzwerk oder einem homogenem IP-Netzerk, abgedeckt werden und die Abrechnungsmodelle überarbeitet werden. Die Menschen wollen immer mehr haben, aber nicht dafür mehr bezahlen. Mit Datenvolumen und Dienstleistungen muss in Zukunft das Geld verdient werden, denn ein fast auf Flatrate basierendes Abrechnungssystem ist jetzt schon kaum mehr bezahlbar für die Telekommunikationsunternehmen.

2 Anwendungsgebiete

2.1 Tracking & Tracing

Beim Tracking und Tracing spricht man in der Regel von der Warenverfolung im Supply Chain. Im einfachsten Fall ist das die Paketverfolung für den Privatmenschen und im Fortgeschritten die Warenverfolgung weltweit in der Luft, im Wasser und auf der Straße. Heutzutage ist es natürlich möglich alle Informationen kontextgefiltert über das Internet jeder Zeit abzurufen.
Mit Tracking und Tracing ist eine absolute Transparenz vom Anfang bis zum Ende der Lieferkette möglich und das in der Regel mit Echtzeit-Informationen. Diese enthalten in der Regel Auftrags-, Lieferanten- und Kundendaten, Packlisten-, Verschiffungs- und Transportinformationen. Somit kann man zu jeder Zeit erfahren, was wann wohin unterwegs ist. Für ein Unternehmen können diese Daten existenzerhaltend sein, wenn Probleme dadurch frühzeitig erkannt werden und ungewollte und teure überraschungen erst gar nicht entstehen können. [Kan07]

Damit diese Daten dann auch wirklich abrufbar sind, müssen sie zu zentralen Servern gelangen. Die Erzeugung geschieht auf der Basis von Barcodes, passiven und aktiven RFID Tags, mit denen die einzelnen Waren, Paletten oder ganze Container ausgerüstet sind. Diese werden an stationären Terminals ausgelesen und via Internet verschickt. Unterwegs auf dem LKW, dem Schiff oder dem Flugzeug übernehmen GSM/GPRS die Kommunikation.

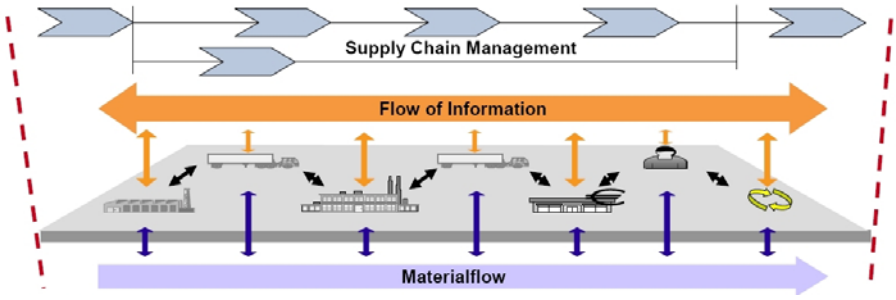


Abbildung 1. Informationsfluss auf dem Weg bis zum Endkunden. [Hom05]

2.2 Identify & Routing

Identify & Routing ist sehr ähnlich zu Tracking und Tracing, weil es im Endeffekt wieder um das Verfolgen bestimmter Dinge geht. In einer Produktionskette können so die Einzelteile bei der Erzeugung identifiziert, mit RFID Tags versehen werden und in den Weiterverarbeitungsschritten jederzeit verfolgt werden. Die Weiterverarbeitung muss nicht immer am selben Ort geschehen, sondern ist

heute oftmals an einem anderen Ort. Werden alle Zwischenproduktschritte zentral in einer Datenbank im Internet zur Verfügung gestellt, können Transporte Zeit optimiert abgewickelt werden oder etwaige Lieferverspätungen genauer kalkuliert werden.
Vorstellen kann man sich auch, dass ein Sicherheitssystem an einem Flughafen darauf ausgerichtet ist, verlorene Koffer aufzufinden oder gar verdächtiges Gepäck zu verfolgen. Voraussetzung wäre, dass jedes Gepäck beim Zugang zum Flughafen einen RFID Tag erhält und somit überall aufgespührt werden kann.

2.3 Categorize

Große Lagerhäuser oder die einfachsten Supermärkte müssen immer über ihre Bestände Bescheid wissen. Das tägliche Zählen der Ware und das damit verbundene Einscannen von Barcodes ist zeitaufwändig und fehleranfällig. Damit ist verbunden, dass immer nur nach diesen Zählungen entsprechende Ware nachbestellt werden kann, was zeitweise unmittelbar zu Engpässen führt. Mit RFID getaggter Ware zählt dieser Misstand zur Vergangenheit. Innerhalb kurzer Zeit kann der ganze Warenbestand elektronisch über RFID-Lesegeräte gezählt werden und am Computer verwaltet werden. Durch Automatisierungsvorgänge kann man soweit gehen, dass beim Auftreten von Engpässen sofort diese Ware über das Internet nachbestellt wird. Anstelle von Personalkosten fallen hier dann die Kosten für RFID-Lesegeräte, die Verkabelung und Software an und das Bestücken jedes einzelnen Stückes Ware.

Ein ganz anderes Problem kommt im Bereich der Operationssäale auf: chirurgische Instrumente. In manchen Krankenhäuser gibt es das davon mehr als 800.000 Stück, die nach jeder OP sterilisiert werden müssen. Die große Gefahr dabei ist, dass niemand weiß, ob diese Instrumente nicht mit einer gefährlichen oder gar tödlichen Krankheit infiziert sind. Mit Hilfe eines eingebetten RFID Tags im Metal des Instrumentes lassen sich nun Informationen speichern, die dem abhelfen. So werden Patientendaten abgelegt, was der Grund und wann die OP war, wann die letzte Sterilisierung war und wie oft dies durchgeführt wurde und vieles mehr. Zieht man in Betracht, dass die Instrumente Krankenhäuser wechseln und extern sterilisiert werden, muss ein Netzwerk (Internet) zwischen den einzelnen Orten vorhanden sein, um ein Management zu ermöglichen. [Gai06]

2.4 Location Aware

Location Aware Anwendungen gehen in Richtung direktem Service beziehungsweise sprechen die Person direkt an.
Auf einem Flughafen in Finnland werden zum Beispiel Informationen per Bluetooth bereitgestellt. Da nicht jeder Handynutzer Bluetooth aktiviert hat, wird er mit einer Push-SMS auf die Bluetooth-Zone hingewiesen, wenn sein Handy sich in die Handynetzwabe des Flughafen anmeldet. Ist nun Bluetooth aktiviert, wird positionsabhängig beziehungsweise in der Reichweite von Bluetoothsendern

Informationen über seinen aktuellen Standort geschickt. Dies können Kaufangebote in der Nähe eines Shops sein oder Flugdaten, die er womöglich selbst zuvor personalisiert hatte.

Wenn wir einen Schritt weitergehen und nun das Handy oder irgendein anderes mobiles Gerät einen RFID Tag hat, so kann man sich vorstellen, dass bald Plakatwände oder andere Werbeflächen den Besitzer direkt ansprechen. Doch hierzu muss die Anzeigefläche mit einer Datenbank kommunizieren und auch die Identität der Person erhalten. Oder trivialer: die Werbefläche „weiß“, wo der Besitzer sich zuletzt aufgehalten hat und kann so entsprechend die Anzeige schalten. Im Kontext einer Einkaufshalle ist dies leicht zu realisieren, wobei man hier zusätzlich noch per SMS/MMS Angebote verschicken könnte, wenn man sich in einem entsprechenden Shop befindet.

Abbildung 2. Links: Kleiderschrank mit RFID-Readern und Monitor[Praa], rechts: Handlesegerät [Prab]

Doch erwünschten digitalen Service kann man heute schon erhalten und zwar im Prada Epicenter Store in Manhattan, New York. Nachdem man sich ein Kleidungsstück ausgesucht hat und in der Kabine anprobiert, werden auf einem großen Touchscreen passende Assecoirs oder passende andere Kleidungsstücke angezeigt (siehe Abbildung 2, links) - ein RFID Reader liest einfach den RFID Chip im Kleidungsstück aus. Man kann dann weiter nach verschiedenen Farben suchen und sich ein ausgewähltes Exemplar schließlich aus dem Lager holen lassen. Ist das noch nicht genug, dann hilft eine Customer Card mit integriertem RFID Chip weiter. In Verbindung mit dieser werden alle bisher gekauften Artikel gespeichert und eine persönliche Homepage steht zur Verfügung. Auf dieser Homepage kann man seine letzte Shoppingtour bequem von zu Hause oder von der Arbeit aus fortführen, um dann zu einem passenden Zeitpunkt die neu aus-

gewälten Kleidungsstücke anzuprobieren. Im Laden direkt sind die Verkäufer mit einem Handlesegerät (siehe Abbildung 2, rechts) ausgestattet, mit denen sie abrufen können, welche Artikel im Lager sind, und gegebenfalls alternative Vorschläge dem Kunden an einem seperaten Monitor präsentieren können. Dieser Service ist bis jetzt lokal beschränkt, doch in Zukunft soll eine Vernetzung aller Shops realisiert werden. So kann man dann an andere Prada Stores in New York weitergeleitet werden oder es ist möglich zu erfahren, was weltweit in anderen Prada Store demnächst an neuer Ware angeboten wird. [Jou02]

Abbildung 3. Links: Mobiles Handgerät mit Standortinformationen, rechts: Blindenleitsystem, [NK06]

Abbildung 4. ucode Tag's [NK06]

In Japan ist man da schon einen Schritt weiter: Anfang 2001 wurde die Initiative „e-Japan Strategy“ von seitens der Regierung und der Industrie gestartet. Diese soll Japan innerhalb der folgenden fünf Jahre zu einer der fortschrittlichsten IT Nationen machen [Buc06]. So wurden in diesem Rahmen zum Beispiel aktive Tags und Sensornetzwerke im ganzen Land eingebettet [NK06]. Sie sollen die Menschen unterstützen, besonders Behinderte und Ausländer (siehe Abbildung

3), doch ebenso Stadtrundfahrten, Shopping oder Lieferungen vereinfachen. Gegenstände und Orte wurden im Rahmen von uID Center mit RFID Tags den sogenannten „ucode Tag's“ versehen (siehe Abbildung 4). Mit Hilfe eines mobilen Gerätes, wie einem Handy oder PDA, kann man die Tags auslesen und per Funknetz zusammen mit dem Kontext beziehungsweise dem Ort die dazugehörigen Informationen abrufen.

2.5 Sensing

Würde ein Käufer mehr Geld bezahlen, wenn er sicher gehen oder es selbst überprüfen könnte, ob Transportbedingungen den Lebensmittel gerecht werden? Zum Beispiel ob die Temperaturrichtlinien und Verfallsdaten eingehalten wurden und er sogar noch Informationen über den Produzenten erfahren kann? In Japan ist das schon möglich. Dort gibt es mittlerweile schon vernetzte Strukturen vom Produzenten bis hin zum Abnehmer im Supermarkt. Hier werden Informationen des Produzenten und des Transportweges in den Supermarkt erfasst. Da besonders auf dem Transportweg oft etwas passieren kann, setzen hier Sensoren ein, wie zum Beispiel Temperatursensoren (siehe Abbildung 5), die den Temperaturverlauf der Ware aufzeichnen. So kann nun der Kunde an speziellen Terminals die Ware einscannen und sich auf einem Monitor alles Wissenswerte anzeigen lassen.

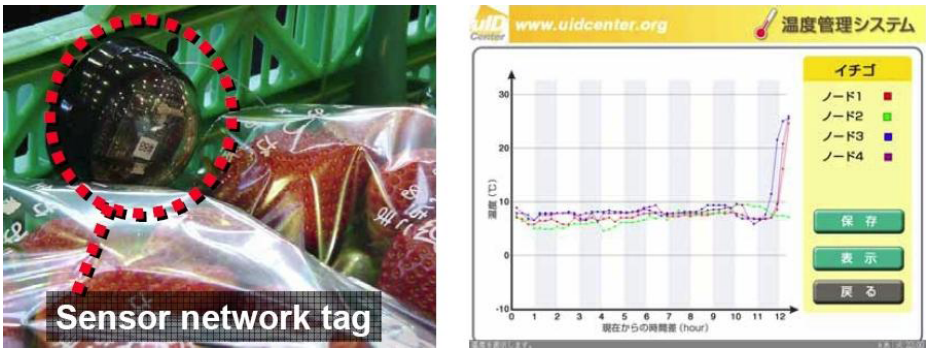


Abbildung 5. Links: Temperatursensor, rechts: Auswertung der Daten, [NK06]

Nicht nur Lebensmittel, sondern den Menschen selbst muss man heutzutage sehr oft überwachen. Besonders die immer älter werdende Gesellschaft ist von vielen Krankheitsbildern gekennzeichnet. So können Sensoren, die das Herz überwachen, den Blutzuckergehalt kontrollieren oder auf andere kritische Werte achten. Es ist denkbar, dass die Sensoren an mobile Geräte angeschlossen sind, die die Daten aufbereiten und bei akuter Gefahr entweder einen Arzt per SMS benachrichtigen oder sogar selbst interagieren und eventuell Medikamente im Körper freisetzen.

Eine ganze andere Dimension bestreitet das Militär: „Smart Dust“ - Kleinstsensoren, die ein „Mesh“-Netzwerk bilden. Sie sind dazu gedacht sich hinter feindlichen Linien unsichtbar in das Gebiet zu legen und Umgebungsinformationen zu sammeln und diese dann per Funk herauszuschaffen. Bisher haben die sogenannten „nodes“ noch den Status von Prototypen. Im Jahr 2004 hatten diese noch die Größe einer Walnuss - heutzutage befinden wir uns im Millimeterbereich (siehe Abbildung 6, rechts oben und unten) - und erreichten ihre Grenzen bei 900 Exemplaren im Netzwerk. Hier besteht voraussichtlich das größte Wachsum in Sachen Internet der Dinge, denn Geld ist reichlich vorhanden. [Sch04b]

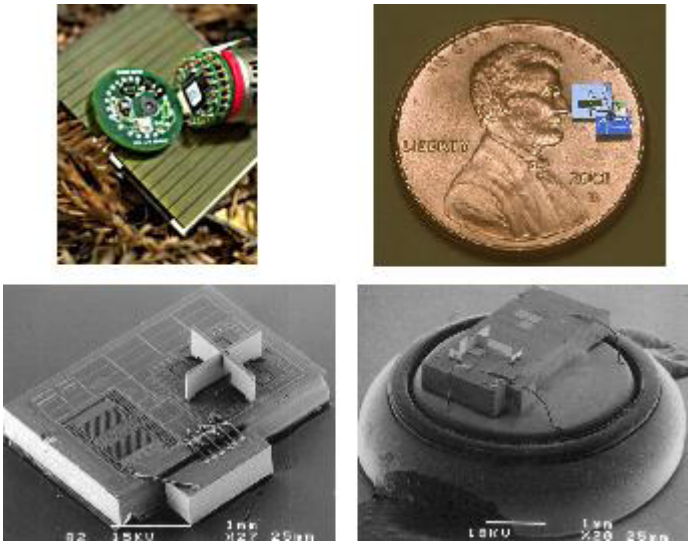


Abbildung 6. Links oben: Umweltsensor [Yan03], Rest: Smart Dust [War04]

Aber nicht nur das Militär setzt auf Kleinstsensoren (siehe Abbildung 6, links oben), sondern auch der Naturschutz. So wurden im Rahmen der californischen Universität Mammutbäume im Mather Redwood Grove mit Sensoren versehen, die Licht, Feuchtigkeit und Temperatur messen. Diese Sensoren sammeln dort hoch in den Bäumen monatelang Daten, die sie per Funk weiterleiten. Vorgesehen ist, dass das Sensornetzwerk auf weitere Waldgebiete ausgeweitet werden soll. [Yan03] Auch im Automobilsport macht die Vernetzung keinen Stop. Was jeder Boxenmechaniker an seinem Monitor betrachtet, wird nun auch dem Zuschauer zu Hause gegen Gebühren angeboten: Telemetriedaten aller Autos in einem Java Applet. In einem virtuellen Amaturenbrett bietet NASCAR.com die Übersicht aller Autos und zeigt in Echtzeit die Geschwindigkeit, Umdrehungszahl, Bremsstärke und Beschleunigung an, basierend auf GPS Telemetrieboxen in den Autos. [Webe][Mel03]

3 Telekommunikation

Die neue Vielfalt an Anwendungen, die durch das Konzept „Internet der Dinge“ entstehen werden, ist schier endlos. Das Datenaufkommen, welches dadurch erzeugt wird, eigentlich unvorstellbar. Nur durch kontinuierlichen Ausbau der Netzwerke und Bereitstellung immer gröerer Datenübertragungsraten wird es überhaupt möglich werden, das Internet der Dinge zu realisieren. Doch wer wird das alles finanzieren? Das Geschäftsmodell der Flatrates mit immer mehr Inklusivdiensten wird dies nicht auffangen können. „Internationale Erfahrungen zeigen, dass der Preislevel trotz steigendem Service Level oft nur gehalten werden kann“ [Fre06a] und Preiserhöhungen vom Kunden nicht akzeptiert werden. D.h. der Kunde will zum Beispiel für die bessere Bandbreite oder für die erweiterten Dienstleistungen nicht mehr zahlen als bisher. So müssen hier ganz neue Wege angestrebt werden, die das Ganze nicht nur zahlbar machen, sondern auch die nötige Sicherheit bereitstellen. Denn in einer Zukunft, in der man fast überall mobile Dienste finden kann, ist eine sichere und bequeme Bezahlung dieser Dienste ein sehr großes Nutzungskriterium. Niemand möchte Opfer von Betrug und Missbrauch seiner Daten werden.

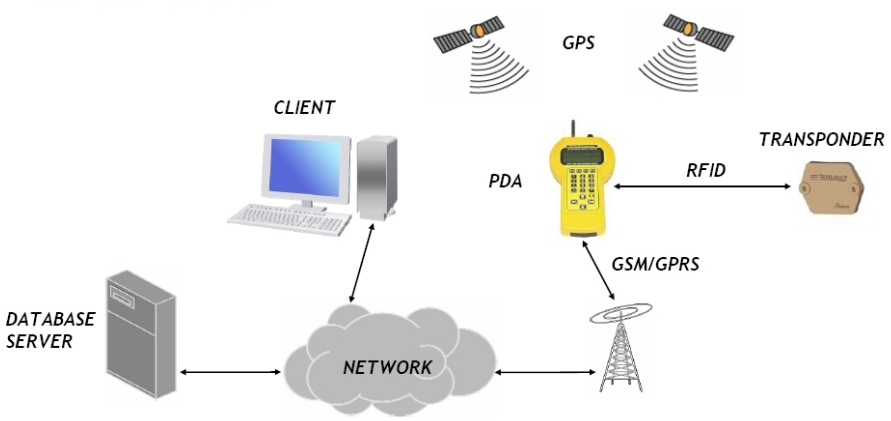


Abbildung 7. Infrastruktur der Telcos: Mobile Netze und das Internet. [Cot06]

3.1 Infrastruktur & Geschäftsmodelle

Die Infrastrukturen der Telekommunikation entwerten sich sehr schnell, daher sind die Telekommunikationsunternehmen schon immer quasi dazu gezwungen Innovationen voranzutreiben. So befinden wir uns gerade in einer Umbruchszeit im Bereich der Mobilfunkstandards. Das schon etwas ältere GMS (General Packet Radio Service) gilt mittlerweile als Auslaufmodell [Hen06], da es durch seine Erweiterung GPRS (Global System for Mobile Communications) fast gänzlich ersetzt wurde. Bis zum Ende dieses Jahrzehntes wird UMTS (Universal Mobile Telecommunications System) dann mehr als die Hälfte des Marktes ausmachen. Mit UMTS wird das Zeitalter der reinen Sprachtelefonie zu ende gehen und die Zeit der weitreichenden Angebote der mobilen Services kommen. Innerhalb der nächsten zehn Jahre werden Funktechnologien wie UWB (Ultra Wide Band) und WiMAX (Worldwide Interoperability for Microwave Access) marktreif sein und noch mehr können. So sollen Übertragungsgeschwindigkeiten von bis zu 500 MBit/s und einem Zelldurchmesser von bis zu 50 km erreicht werden und ganz neue Möglichkeiten bieten, die besonders dem Internet der Dinge in die Hände spielen wird [Hen06]; zum Beispiel sollen mit dieser (WiMAX) Technik breitbandige Zugänge zum Internet via Funknetz angeboten werden und „always on“ im mobilen Sinn wird realer denn je! [Wik07b]

Zukunftsmusik ist toll, bringt aber kein Geld. Gerade Privatkunden besitzen bei ihrem Breitbrandanschluss entweder Volumentarife oder Flatrates, diese werden in der Geschwindigkeit regelmäßig verbessert bei gleichbleibenden Gebühren. Somit sind gerade Flatrates keine weitere Einnahmequelle und sind nach der Abschöpfung nicht mehr rentabel, da der Wettbewerb die Preise nach unten drückt.

Im Bereich des Handys ist die SMS der meistgenutzte Dienst schlechtweg hin. Aber nicht nur zum Mitteilungen schreiben ist die SMS nutzbar, sondern auch zum Bezahlen. So hat sich ein gewaltiger Markt für Klingeltönen, Logos und Spiele für das Handy etabliert. Ein deutsches Marktforschungsinstitut prognostiziert für das Jahr 2007 eine Steigerung von gut 300% im Bereich der mobilen Unterhaltung und Handys [Gen]. In diesem Kontext ist der Marktführer Jamba zu nennen, der seit dem Gründungsjahr 2000 diesen Markt bedeutend mit aufgebaut hat und in der Zukunft besonders im Bereich UMTS aktiv werden will.

Ein weitreichendes Problem gerade in Deutschland ist, dass die Handybesitzer nicht einmal wissen, was sie mit ihrem mobilen Gerät machen können. Das hat zur Folge, dass die Funktionen nicht genutzt werden und etwaige Anwendungen kein Geld einbringen und wieder eingestellt werden müssen. Hier könnten gezielten Werbekampagnen den Besitzern zeigen, welchen Mehrwert sie mit diesen Geräten erzielen könnten. Denn genau darin liegt ein großes Problem: viele Anwendungen oder mobile Dienste können sich nicht etablieren, da sie nicht bekannt sind, keinen spürbaren Mehrwert für den Besitzer erbringen im Gegenzug dafür, dass er bezahlen muss, oder einfach nur die Erfolgsfaktoren (stabil, ob-

jektiver/fühlbarer Nutzen, aktuell und qualitativ, einfaches/transparents Preismodell, serios/sicher) erfüllen [JK06].
Ein gutes Beispiel sind die Verkehrsbünde in Karlsruhe [Webc] und München [Webd], die entweder online oder offline ihre Fahrpläne als Java-Anwendung kostenlos auf ihren Homepages oder als kostenpflichtigen Download für das Handy anbieten. Aber es sind nicht nur statische Daten schon verfügbar, sondern im Falle Karlsruhe sogar schon Echtzeitdaten [Webb] im Internet abrufbar. Werden diese in Zukunft als mobiler Dienst angeboten, so kann man nach Belieben zusammen mit Positionsinformation erfahren, wann welche Straßenbahn in der Umgebung vorbeifährt.

Ein großes Manko, welches auch eine große Einschränkung mit sich bringt, ist die Heterogenität der Netze. So sind Bluetooth, GSM und WLAN inkompatible zueinander. Bei Bluetooth und WLAN ist es im Gegensatz zu GSM nicht möglich die Zellen eines Senders zu wechseln, ohne dass die Verbindung unterbrochen wird - ein wirklich mobiles Gefühl mag da nicht aufkommen. Aber die Entwicklungen gehen in Richtung eines konvergenten Netzes, was auf IP basieren wird [Kna06]. Dann wird es keine bestimmten Anwendungen und Geräte für bestimmte Netze geben, sondern jede Anwendung und jedes Gerät wird im gesamten Netz lauffähig sein.
Seit letztem Jahr ist in dieser Richtung schon etwas mit der Einführung von Triple-Play passiert. Darunter versteht man „das gebündelte Angebot verschiedener Dienste“ wie TV/Video, Internet/Daten und Sprach-Telefonie [Fre06b], die man von nur noch einem Anbieter bezieht. Wenn später noch der Mobilfunk hinzukommen sollte wird daraus Quadruple-Play. Möglich machten diesen Zusammenschluss vor allem die hohe Breitbandabdeckung, der Konkurenzdruck infolge von Infrastrukturausbau und die Akzeptanz der Kunden speziell im Bereich TV/Video für Inhalte zu zahlen.

Abgesehen von den verfügbaren Netzen und deren Protokollen ist im Bereich der Endgeräte, wie Handys, PDAs, PocketPC's und weiteren GPRS fähigen Geräten, eine neuer Technologieschritt möglich. Verfügen diese Geräte nämlich über NFC (Near Field Communication) Technologie, so kann diese Funktechnologie dazu dienen, mit allen möglichen Dingen in nächster Nähe (ca. 10cm Reichweite) zu kommunizieren - aktiv oder passiv. So können Geräte mit NFC zum Beispiel als Zahlungsmittel, Fahrkarte im öffentlichen Verkehr, als Schlüssel für Gebäude oder als Lesegerät von entsprechenden Info-Spots dienen.
Denkbar wäre auch folgendes Szenario. Um an Fußballticktes zu kommen wählt man mit seinem Handy oder ähnlichem Gerät eine Anwendung, die es via GPRS ermöglicht, Tickets zu kaufen. Zuerst muss man seine UserId mit seinem Passwort verifizieren, nun kann man die Tickets kaufen und erhält die Bestätigung. Am Spieltag geht man zum Stadion und streift mit dem Handy im geringem Abstand über das Einlasstor, was dieses öffnet und einen Sitzplan auf das Handy sendet. D.h. das Telekommunikationsunternehmen fungiert als Trusted Third Party, deren Aufgabe darin, besteht Geschäfte für andere Firmen durchzuführen.

Das behinhaltet auch das Indentifizieren der Kunden und eine sichere Bezahlung. Das Telekommunikationsunternehmen verdient dann daran, dass sie als Trusted Third Party ins Spiel kommt und die Nutzer die Infrastruktur und Dienste der Telekommunikationsunternehmen benutzen.

Wird sich dies durchsetzen, so ist mit einer neuen Flut von Datenaufkommen zu rechnen im Bereich von GPRS und SMS. Zusätzliche werden Gebühren für die Transaktionen des Bezahldienstes und der Erwerb eines NFC fähigen Gerätes fällig. So werden hier viele neue Möglichkeiten entstehen, mit denen sich viel Geld verdienen lässt. [Mei06]

4 Fazit

Eins ist klar, wer beim Internet der Dinge mitmacht, verliert an informationeller Selbstbestimmung, d. h. jeder einzelne hat, nun ob er will oder nicht, durch die „versteckten“ Sender und Transponder keinen Einfluss mehr darauf, welche Informationen preisgegeben werden [Wik07a].
Ersten Widerstand gibt es bereits gegen die „unsichtbare“ Integration der neuen Technologien, denn sie macht schon heute vielen Menschen Angst, wie zum Beispiel die Kampagne „Boycott Bennetton - No RFID tracking chips in clothing!“ [Weba] zeigt. So hat Bennetton vor, passive RFID Tags in ihre Begleitungsartikel zu integrieren, um logistische Abläufe zu verbessern. Jedoch ist eine Entfernung der Tags nicht vorgesehen. Die Menschen bangen um ihren Rest von Privatheit, da diese Tags jederzeit von jedem ausgelesen werden können. Diese Kampagne scheint aber wiederum die Einführung und Weiterentwicklung dieser neuen Technologien zu bremsen, denn die Verantwortlichen gingen bereits einen Schritt zurück und distanzierten sich von ihrem Vorhaben. Ein Ausweg hier wäre die Zerstörung der Tags nach dem Kauf der Ware.
Das weit verbreitete Rabattsystem „Payback“ [Webf] verbirgt Intentionen, die vielen unbekannt sind. So sammelt diese unscheinbare Karte bei seinen über 60 Partnern Informationen über den Besitzer, wenn er sie einsetzt. Dies repräsentiert eine neue Art der Marktforschung, wie sie noch nie dagewesen ist. Wer seine Kaufgewohnheiten nicht preisgeben will, muss somit indirekt dafür bezahlen, da ihm keine Rabatte gewährt werden können.

Bis jedoch die große Verbreitung von RFID Tags im Alltag eingeführt wird, muss erst die Massenproduktion die Stückpreise der Tags erheblich senken. Helfen könnten neue Ideen, die Tags aus neuen Materialien (z. B. auf Polymerbasis) ermöglichen und/oder aufdruckbar machen.
Ein großes Problem ist der Missbrauch der neuen Technologien. Passive RFID Tags sind jederzeit von jedem mit der entsprechenden Hardware unkontrolliert auslesbar. Nicht nur das ungewollte Auslesen, sondern auch das „Imitieren“ von Tags stellt eine Herausforderung da. So kann man das Signal auffangen und zu einem späteren Zeitpunkt wieder senden, der Empfänger kann nicht unterscheiden, wer ihm da gerade antwortet. Abhilfe schaffen könnten moderne RFID-Tags, die

mit Protokollen arbeiten und eine Verschlüsselung anwenden.
Aber hinter allen Anwendungen ist es der Datenaustausch, der erst das Internet der Dinge bildet. Ohne weltweiten Informationsaustausch gibt es keinen Mehrwert. Immer mehr Bandbreite müssen die Telekummikationsunternehmen zur Verfügung stellen, um der Nachfrage ihrer Kunden nachzukommen. Der hohe Konkurrenzdruck aber drückt die Preise in den Keller. Man wird in Zukunft wohl auf andere Bezahlungen umsteigen müssen, als rein nach Datenaufkommen oder mit Flatrates abzurechnen. So wäre es möglich nicht mehr für das Datenaufkommen, sondern für den Service oder den Inhalt direkt zu bezahlen, wobei die Telekommunikationsbranche sozusagen Transaktionensgebühren erhebt. Bringt man das mit neuen Ideen wie der elektronischen Brieftasche im Handy oder Notebook zusammen, dann würde man eine neue Art der Bezahlung einführen für die Welt des Internet der Dinge. Das Serven im Internet am Flughafen, das Drucken auf öffentliche Drucker, das Kaffetrinken um die Ecke oder das Abfragen des heimischen Sicherheitssystemes würde über die gleiche Art und Weise abgerechnet werden. [PBW03]

Bis alles jedoch soweit ist sind noch einige Hürden zu überwinden. Es werden Standardisierungen nötig sein, damit nicht jeder seine eigene Lösung durchsetzen will. Das Vertrauen in die Menschen, den Benutzern der neuen Technologien, muss gewonnen werden, indem man den Umgang sicher gestaltet und die Technik nicht gegen einen eingesetzt werden kann. Anders wie in Japan müssen die Investoren und Industrie die Kosten selbst tragen. Denn auf der anderen Seite der Welt steht die Regierung mit einem Modell dahinter und subventioniert die Technologisierung des eigenen Landes weitesgehenst.

Literatur

Buc06. Buckley, John. *FROM RFID TO THE INTERNET OF THINGS*. http://www.rfidconsultation.eu/docs/ficheiros/WS_1_Final_report_27_Mar.pdf. 2006

Cot06. Cottino, Edoardo. *RFID for telecommunication network maintenance*. http://www.itu.int/ITU-T/worksem/rfid/presentations/s2p2cottino.zip. 2006

Fre06a. Freyberg, Axel: *„Triple-Play" – Geschäftsmodelle der Zukunft*. http://www.hessen-it.de/mm/VortragFreyberg.pdf. 2006. – Seite 7

Fre06b. Freyberg, Axel: *„Triple-Play" – Geschäftsmodelle der Zukunft*. http://www.hessen-it.de/mm/VortragFreyberg.pdf. 2006. – Seite 2

Gai06. Gaisch, Robert. *RFID Solutions for e-health*. http://www.itu.int/ITU-T/worksem/rfid/presentations/s2p5gaisch.zip. 2006

Gen. Genzmer, Niels: *Jamba The Portal for Digital Entertainment*. http://ad.jamba.de/help/N_EN_Jamba_Portrait_mitGrafik_DE.pdf. – [Online; Stand Januar 2007]

Hen06. Heng, Dr. S. *UMTS - eine schleichende Erfolgsgeschichte?* http://www.ecin.de/mobilebusinesscenter/umts-realitaet. 2006

Hom05. ten Hompel, Prof. M. *Das Internet der Dinge wird die Welt bewegen*. http://www.vdeb.de/download/Logistik_morgen_Vortrag_ten_Hompel.pdf. 2005

JK06. rer. nat. Jrgen Kaack, Dr. *mCommerce, erfolgreiche Vermarktung und Fazit*. http://www.ecin.de/mobilebusinesscenter/entwicklungen/index-2.html. 2006

Jou02. Journal, RFID: *Learning from Prada*. http://www.rfidjournal.com/article/articleview/272. 2002. – [Online; Stand Januar 2007]

Kan07. Kanow, Andreas. *See Change Services*. http://www.apllogistics.de/de/service/See_Change_Services.php. 2007

Kna06. Knauer, Peer. *Zukunft Telekommunikation*. http://www.hessen-it.de/mm/VortragKnauer.pdf. 2006

Mei06. Meindl, Reinhard. *Experience simplicity with Near Field Communication (NFC)*. http://www.itu.int/ITU-T/worksem/rfid/presentations/s0p7meindl.zip. 2006

Mel03. Meloan, Steve: *Toward a Global „Internet of Things"*. http://java.sun.com/developer/technicalArticles/Ecommerce/rfid/. 2003. – [Online; Stand Januar 2007]

NK06. Noboru Koshizuka, Ph. D. *Future trends in Japanese RFID Technologies and Market*. http://www.itu.int/ITU-T/worksem/rfid/presentations/s6p4koshizuka.zip. 2006

PBW03. P. Boddupalli, N. Davies A. Friday O. S. ; Wu, M. *Payment Support in Ubiquitous Computing Environments*. http://ieeexplore.ieee.org/iel5/8785/27816/01240772.pdf?tp=&arnumber=1240772&isnumber=27816. 2003

Praa. *Case Studies: PRADA: RFID Closet*. http://www.ideo.com/case_studies/prada.asp?x=5

Prab. *Case Studies: PRADA: Staff Device*. http://www.ideo.com/case_studies/prada.asp?x=3

Sch04a. Schmitt, Stefan. *Die dritte Revolution*. http://www.heise.de/tr/artikel/50287/2/0. 2004

Sch04b. SCHMITT, Stefan. *Die dritte Revolution*. http://www.heise.de/tr/artikel/50287/4/0. 2004

War04. WARNEKE, Brett. *Smart Dust*. http://www-bsac.eecs.berkeley.edu/archive/users/warneke-brett/SmartDust/index.html. 2004

Weba. *Boycott Benetton – No RFID tracking chips in clothing!* http://www.boycottbenetton.com

Webb. *init – innovation in traffic systems AG*. http://www.init-ka.de/webfgi/query.aspx?navid=12

Webc. *Karlsruher Verkehrsverbund*. http://kvv.de/kvv/fahrplan/handyfahrplan/start.php?navid=13

Webd. *Münchner Verkehrs- und Tarifverbund*. http://www.mvv-muenchen.de/de/home/fahrgastinformation/efa/mobiledienste/index.html

Webe. *NASCAR.COM: TrackPass*. http://www.nascar.com/multimedia/index.html

Webf. *PAYBACK*. http://www.payback.de

Wik07a. WIKIPEDIA: *Radio Frequency Identification — Wikipedia, Die freie Enzyklopädie*. http://de.wikipedia.org/w/index.php?title=Radio_Frequency_Identification&oldid=26267972. 2007. – [Online; Stand Januar 2007]

Wik07b. WIKIPEDIA: *WiMAX — Wikipedia, Die freie Enzyklopädie*. http://de.wikipedia.org/w/index.php?title=WiMAX&oldid=26236945. 2007. – [Online; Stand Januar 2007]

Yan03. YANG, Sarah. *Redwoods go high tech: Researchers use wireless sensors to study California's state tree*. http://www.berkeley.edu/news/media/releases/2003/07/28_redwood.shtml. 2003

Aktuelle RFID-Protokolle und ihre Einsatzmöglichkeiten im Einzelhandel

Stefan Minden

1. Einführung

Mit Hilfe der RFID-Technologie können im Einzelhandel und der Einzelhandelslogistik zahlreiche neue Anwendungen ermöglicht oder traditionelle Geschäftsvorfälle vereinfacht werden. Dies gilt besonders bei Kennzeichnung einzelner Produkte mit RFID-Tags, dem so genannten Item Level Tagging.
In dieser Arbeit sollen die Anforderungen und Erfahrungen der Branche mit RFID zusammengestellt und mit den aktuellen technischen Möglichkeiten von RFID-Systemen abgeglichen werden. Der Fokus wird dabei auf für das Item Level Tagging wichtige Leistungsmerkmale gelegt. Des Weiteren wird untersucht, auf welchen Wegen die Leistung der Technologie verbessert werden kann, um den Einsatz des Item Level Taggings zu begünstigen.
In Kapitel 2 werden zunächst mögliche Einsatzszenarien von RFID im Einzelhandel und der Einzelhandelslogistik identifiziert. Zugleich wird untersucht, auf welcher Kennzeichnungsebene die Güter jeweils mit RFID-Tags versehen sein müssen, um die Umsetzung der Szenarien möglich zu machen. Für die Szenarien kritische Leistungsmerkmale werden ebenfalls behandelt.
In Kapitel 3 wird die Funktionsweise von RFID-Systemen erklärt. Des Weiteren werden Möglichkeiten zur Gestaltung der Systeme zusammengestellt und ihre Auswirkungen auf die Leistung ermittelt.
In Kapitel 4 werden die Ergebnisse einer Umfrage unter deutschen Einzelhandelsunternehmen zu ihrem Umgang und ihren Erfahrungen mit RFID-Technologien präsentiert. Das Ziel der Umfrage bestand darin, herauszufinden, welche Szenarien von den Unternehmen mit welcher Technologie bearbeitet werden und welche Erfolge oder Probleme sie dabei feststellen.
In Kapitel 5 werden Möglichkeiten zur Leistungssteigerung von RFID-Systemen betrachtet. Diese liegen in

- der Auswahl des Protokolls für die Luftschnittstelle, verbunden mit der Entscheidung für eine bestimmte Betriebsfrequenz,
- der Wahl leistungsfähiger Zugriffsverfahren als Teil des Protokolls sowie
- der Ausnutzung der durch das Datenformat bereitgestellten Möglichkeiten.

Ein Fazit wird schließlich in Kapitel 6 gezogen.

2. Einsatzszenarien in Einzelhandel und Einzelhandelslogistik

Die Szenarien lassen sich in zwei Gruppen einteilen:

- Geschäftsvorfälle in Logistik und Lagerhaltung
- Anwendungen im Verkaufsraum beim Umgang mit dem Kunden

Ihnen ist gemeinsam, dass ein RFID-Einsatz zu einer Automatisierung in Verbindung mit einer Beschleunigung, einer Verringerung der Fehlerhäufigkeit und einer Reduzierung des Personalbedarfs führen kann. Einige der Szenarien werden heute mit Hilfe von Barcodes bearbeitet.

2.1 Szenarien in Logistik und Lagerhaltung

Zu den Geschäftsvorfällen in Logistik und Lagerhaltung zählen (vgl. [metro-ic]):

- Sendungsverfolgung: Es besteht die Möglichkeit, jederzeit nachzuvollziehen, an welchem Ort in der Logistikkette sich ein Produkt momentan befindet. Eine Steuerung der Logistik wird dadurch vereinfacht. In Distributionszentren kann die Ware automatisch sortiert und dem Zielort entsprechend verteilt werden.
- Wareneingang: Ein automatischer Abgleich von Lieferschein und angelieferter Ware kann durchgeführt werden. Bei Entwicklung vollautomatisierter Wareneingangstore kann die Anlieferung auch ohne Anwesenheit von Lagerpersonal rund um die Uhr erfolgen.
- Warenausgang: In Distributionszentren kann die Kommissionierung der Ware beschleunigt und die Fehlerhäufigkeit verringert werden.
- Umlagerung: Die Lagerstelle eines Gutes kann automatisch bestimmt werden. Auf lange Sicht wäre sogar eine automatisierte Umlagerung denkbar. Des Weiteren kann eine Umlagerung aufgrund zusätzlich gespeicherter Attribute wie des Mindesthaltbarkeitsdatums angestoßen werden.
- Inventur: Eine automatische Feststellung des aktuellen Lagerbestandes kann jederzeit auf Knopfdruck durchgeführt werden. Durch Abgleich mit den Soll-Werten für die Lagerbestände wird der Schwund schnell deutlich.
- Nachbestellungen: Bei Unterschreiten einer gewissen Grenze an vorhandenen Artikeln kann automatisch eine Nachbestellung ausgelöst werden, deren Größe z.B. anhand der beobachteten Nachfrage oder abhängig von der Haltbarkeitsdauer berechnet wird. Auch der Grenzwert kann entsprechend der festgestellten Nachfrage vom System bestimmt werden.

Mögliche Kennzeichnungsebenen sowie notwendige Leistungsmerkmale gehen aus Tabelle 1 hervor.

Szenario	**Kennzeichnungs-ebenen**	**Zuverlässigkeit**	**Reichweite**	**Leserate**
Sendungs-verfolgung	alle	nahe 100%	ca. 2m	ca. 100/s
Wareneingang	alle	nahe 100%	ca. 2m	ca. 100/s
Warenausgang	alle	nahe 100%	ca. 2m	ca. 100/s
Umlagerung	alle	nahe 100%	ca. 2m	ca. 100/s
Inventur	Item Level	nahe 100%	ca. 1m	>> 100/s
Nachbestellungen	Item Level	nahe 100%	ca. 1m	>> 100/s

Tabelle 1: Mögliche Kennzeichnungsebenen und notwendige Leistungsmerkmale zur Umsetzung der Logistik-Szenarien

Bei den ersten vier Szenarien kann eine Kennzeichnung sowohl auf Paletten- als auch auf Packstück- oder Einzelproduktebene erfolgen. Bei Kennzeichnung größerer Einheiten sind allerdings die Leistungsanforderungen an das RFID-System, insbesondere an die Zahl lesbarer Tags pro Zeiteinheit, geringer. Bei den letzten beiden Szenarien

ist eine präzise Bestimmung des Lagerbestandes nur auf Basis des Item Level Taggings möglich.
Eine hohe Zuverlässigkeit ist prinzipiell bei allen Szenarien anzustreben. Bei der Inventur kann sie beispielsweise unter der Verwendung von Metallregalen leiden, die die Signale des Lesegerätes absorbieren oder reflektieren.
Bezüglich der Reichweite müssen bei den ersten drei Szenarien Warenein- und -ausgangstore überbrückt werden können, so dass ein Wert von wenigen Metern nötig ist. Bei der Umlagerung müssen Lagerplätze mit RFID-Lesegeräten versehen sein, die ebenfalls eine Reichweite von wenigen Metern aufweisen, um Paletten vollständig zu erfassen. Bei Inventur und Nachbestellungskontrolle sind Verkaufsregale mit Lesegeräten zu bestücken. Falls diese an Regalböden befestigt sind, genügt eine Reichweite von etwa einem Meter. Größere Reichweiten sind jeweils sogar als schädlich zu bewerten, da so eine größere Transponderzahl in den Ansprechbereich geraten kann, was sich negativ auf die Zuverlässigkeit auswirkt und höhere Ansprüche an das Zugriffsverfahren stellt.
Die Bedeutung der Leserate hängt von der verwendeten Kennzeichnungsebene ab. Insbesondere bei Inventur und Nachbestellungskontrolle müssen wegen des Item Level Taggings so hohe Leseraten möglich sein, dass auch einige hundert Artikel auf einem Regalboden in annehmbarer Zeit erkannt werden können. Aber auch bei den ersten vier Szenarien können durchaus Leseraten im dreistelligen Bereich nötig sein, um alle Packstücke identifizieren zu können, die gleichzeitig ein Warentor passieren.

2.2 Szenarien im Verkaufsraum

Mögliche Anwendungen im Verkaufsraum, die sich auch auf den Umgang mit dem Kunden auswirken, sind (vgl. [metro-ic]):

- Preisanzeige: Kleine Displays am Regal können den mit dem Artikel verbundenen Preis anzeigen. Somit wäre es möglich, Aktionspreise automatisch einzuführen, ohne dass dazu eine manuelle Bearbeitung von Preisschildern notwendig wäre.
- Retoure: Das Erfassen von zurückgegebenen Waren, z.B. von Leergut, kann automatisiert werden. Bei dieser Gelegenheit können auch Eigenschaften wie das Mindesthaltbarkeitsdatum oder das Alter geprüft werden, das beispielsweise im Garantiefall eine Rolle spielt. Ein Vorlegen der Einkaufsquittung durch den Kunden wäre nicht mehr nötig, da vom System festgestellt werden kann, wann und wo das Produkt gekauft wurde.
- Warensicherung: Wie heute bereits bei einigen Produkten üblich können Transponder zum Schutz vor Diebstahl verwendet werden. Die Ausweitung dieses Ansatzes auf das gesamte Sortiment wird ermöglicht.
- Kasse: Eine automatisierte Kasse ist langfristig vorstellbar. Dabei müssen nicht mehr alle Artikel einzeln und manuell gelesen werden, sondern pulkweise vom System. Eine Aufhebung des Diebstahlschutzes kann im gleichen Atemzug vorgenommen werden.
- Informationsversorgung des Kunden: Auf dem Transponder gespeicherte Hintergrundinformationen zum Produkt können vom Kunden bei Bedarf abgerufen werden, beispielsweise in einer intelligenten Umkleidekabine. Intel-

ligente Einkaufswagen mit eingebautem RFID-Lesegerät erfassen alle darin befindlichen Produkte und zeigen jederzeit den Gesamtpreis an.

Mögliche Kennzeichnungsebenen und notwendige Leistungsmerkmale sind in Tabelle 2 aufgeführt.

Szenario	**Kennzeichnungs-ebenen**	**Zuverlässigkeit**	**Reichweite**	**Leserate**
Preisanzeige	Item Level	nahe 100%	ca. 1m	>> 100/s
Retoure	Item Level	nahe 100%	<< 1m	<< 100/s
Warensicherung	Item Level	nahe 100%	ca. 2m	>> 100/s
Kasse	Item Level	nahe 100%	ca. 1m	>> 100/s
Informations-versorgung	Item Level	nahe 100%	<< 1m	<< 100/s

Tabelle 2: Mögliche Kennzeichnungsebenen und notwendige Leistungsmerkmale zur Umsetzung der Verkaufsraum-Szenarien

Da die Produkte im Verkaufsraum nicht mehr auf Paletten oder in Packstücken gelagert werden, ist eine Kennzeichnung auf Einzelproduktebene nötig.
Die Anforderungen an die Zuverlässigkeit sind auch hier hoch. Metallische Regale können bei der Preisanzeige oder metallische Einkaufswagen bei der elektronischen Kasse für Probleme sorgen.
Bei Reichweite und Leserate gelten für die Preisanzeige die gleichen Voraussetzungen wie schon bei der Inventur. Bei Kasse und elektronischem Einkaufswagen sollte ein Einkaufswagen vollständig erfasst werden können, so dass eine Reichweite von gut einem Meter benötigt wird. Da in einem Einkaufswagen durchaus eine dreistellige Produktzahl transportiert werden kann, ist auch die Leserate von großer Bedeutung. Gleiches gilt bei der Warensicherung. Hierfür werden an den Ausgängen Lesegeräte mit Reichweiten von etwa zwei Metern benötigt. Bei Retoure und der elektronischen Umkleidekabine sind die Reichweiten von geringer Bedeutung, da Artikel hier direkt an ein Lesegerät gehalten werden können. Für diese beiden Szenarien ist die Leserate unwichtig, weil die Produkte einzeln erfasst werden.

3. Grundlagen zu RFID-Systemen[1]

3.1 Aufbau und Funktionsweise

RFID-Systeme bestehen grundsätzlich aus Lesegeräten und Transpondern, die auch als RFID-Tags oder -Etiketten bezeichnet werden und an den zu kennzeichnenden Produkten befestigt sind. Die Lesegeräte werden an Anwendungssysteme angeschlos-

[1] [finkenzeller]

sen, die für eine Verknüpfung der gelesenen Daten mit weiteren in Datenbanken hinterlegten Informationen sorgen (siehe Abbildung 1).

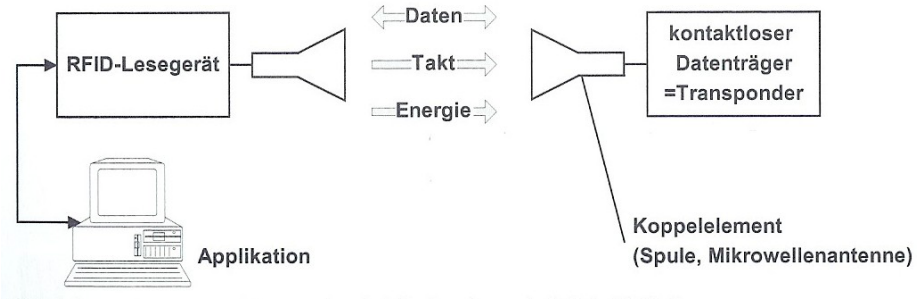


Abbildung 1: Funktionsweise eines RFID-Systems (entnommen aus [finkenzeller])

Ein Transponder – der Ausdruck setzt sich aus den Begriffen Transmitter und Responder zusammen [wikipedia-transponder] – besteht dabei aus einem Mikrochip und einem Kopplungselement wie einer Spule oder Antenne. Während auf dem Chip die zur Identifikation notwendigen Daten gespeichert sind, dient das Kopplungselement dem Empfang der vom Lesegerät ausgesendeten Energie und des Taktes.
Die Erscheinungsformen der Transponder sind vielfältig. Sie können beispielsweise in runde Kunststoffgehäuse mit geringem Durchmesser eingebaut werden (so genannte Disks). Weitere gängige Bauformen sind Chipkarten oder Smart Labels, bei denen der Transponder auf sehr dünnen Etiketten untergebracht ist.
Auch bei den Lesegeräten sind diverse Bauformen vom Handheld-Gerät bis zu stationären Lösungen im Einsatz. Das Gerät selbst besteht aus einer Steuerungseinheit und einer Hochfrequenz-Schnittstelle und verfügt ebenso wie die Transponder über ein angeschlossenes Koppelelement.
Betrachtet man die Funktionsweise, wird deutlich, dass das Lesegerät bei der Energieversorgung des Transponders eine entscheidende Rolle spielt. Befindet sich der Transponder außerhalb der Reichweite eines Lesegerätes, ist er – selbst wenn er eine eigene Batterie besitzt – nicht in der Lage, ein eigenes Signal zu senden. Gerät er jedoch in ein von einem Lesegerät aufgebautes magnetisches oder elektromagnetisches Feld, wird in der Transponderspule ein Strom induziert, der den Mikrochip aktiviert.
Die Datenübertragung zum Lesegerät erfolgt nun durch eine Veränderung des vom Lesegerät erzeugten Feldes. Ein Verfahren hierzu ist die Lastmodulation, die im Niedrig- und Hochfrequenzbereich bei induktiver Kopplung eingesetzt wird. Hierbei erkennt das Lesegerät das Ausmaß, in dem der Transponder dem magnetischen Feld Energie entzieht. Durch Variation dieser Größe mit Hilfe eines Lastwiderstandes kann der Transponder auf diese Weise Daten an das Lesegerät übertragen. Bei der Backscatter-Kopplung im Ultrahochfrequenzbereich wird ein elektromagnetisches Feld eingesetzt. Die vom Lesegerät abgestrahlte Leistung wird in diesem Fall an der Transponderantenne teilweise reflektiert. Dabei kann mit Hilfe eines Lastwiderstandes und abhängig von den zu übertragenden Daten die Amplitude moduliert werden.

Auch über die Subharmonische der Betriebsfrequenz des Lesegerätes kann eine Datenübertragung erfolgen.

3.2 Gestaltungsmöglichkeiten

3.2.1 Frequenz

Besonders weitreichende Folgen auf die Leistungsfähigkeit und die Einsatzmöglichkeiten eines RFID-Systems hat die vom Lesegerät gesendete Betriebsfrequenz.
Sie lässt sich klassifizieren in den Niedrigfrequenzbereich LF (low frequency) von 30 bis 300 kHz, den Hochfrequenzbereich HF (high frequency) von 3 bis 30 MHz und den Ultrahochfrequenzbereich UHF (ultra high frequency) von 300 MHz bis 3 GHz. Darüber hinaus gehende Frequenzen liegen im Mikrowellenbereich.
Mit höheren Frequenzen können größere Reichweiten und Datenübertragungsraten erzielt werden. Allerdings treten im Gegenzug Probleme mit Reflexion und Absorption bei metallischen Gegenständen sowie die Störungen der Signale durch Flüssigkeiten auf [gs1-aut, wikipedia-rfid], die für eine Verringerung der Reichweiten und eine Verschlechterung der Zuverlässigkeit sorgen.
Eine besondere Problematik ergibt sich im UHF-Bereich. Wegen der Zuständigkeit nationaler Behörden für die Freigabe von Frequenzen sind für RFID-Anwendungen in Europa und den USA verschiedene Frequenzbereiche vorgesehen: in Europa 865-868 MHz, in den USA 902-928 MHz. Immerhin ist es trotz dieser Unterschiede möglich, in Europa beschriebene Chips in den USA wieder auszulesen [gs1-faq]. Die Frequenzdifferenz äußert sich aber auch in geringeren Reichweiten und Erkennungsraten der europäischen Systeme [metro-frequenz, alien]. Die Standardisierungsorganisation GS1 Germany weist darauf hin, dass mittlerweile die Funkregularien in Europa angepasst werden, so dass durch eine verbesserte Kanalgestaltung vergleichbare Werte erreicht werden können [füßler].

3.2.2 Energieversorgung

Auch wenn Transponder wie oben beschrieben die zur Aktivierung des Chips notwendige Energie dem magnetischen oder elektromagnetischen Feld des Lesegerätes entziehen, besteht die Möglichkeit, eine zusätzliche Energieversorgung in Form einer Stützbatterie einzubauen. Die von ihr zur Verfügung gestellte Energie dient der Versorgung des Chips, so dass ein relativ schwaches Feld zum Betrieb des Transponders ausreicht. Zur Datenübertragung wird dieses jedoch weiterhin benötigt, da auch mit Stützbatterie kein eigenes Signal vom Transponder ausgehen kann. Mit Hilfe der Batterie können so zwar größere Reichweiten ermöglicht werden, die aber kaum über wenige Meter hinausgehen. Transponder mit eigener Energieversorgung werden als aktiv oder semi-passiv bezeichnet, Transponder ohne Stützbatterie als passiv.
Von Nachteil sind die höheren Preise für aktive und semi-passive Transponder. Gerade beim Item Level Tagging ist der Transponderpreis ein wichtiges Kriterium. Besonders große Reichweiten werden ohnehin nicht benötigt, so dass im Einzelhandel eher passive Tags Verwendung finden.

3.2.3 Speicherkapazität und Beschreibbarkeit

Die möglichen Speicherkapazitäten der Transponder reichen von einem Bit bis zu vielen Kilobytes. 1-Bit-Transponder werden vor allem zum Diebstahlschutz eingesetzt, da hier nur interessiert, ob sich der Transponder in naher Umgebung zum Lesegerät befindet oder nicht.
Zu Identifikationszwecken werden häufig nicht-beschreibbare Transponder mit Speicherkapazitäten von einigen Byte eingesetzt. Zur Kodierung des Electronic Product Code (EPC) als Datenformat zur eindeutigen Identifikation im Warenverkehr werden je nach Variante 64 oder 96 Bit benötigt. Die mit dem populären Standard EPCGlobal Class 1 Generation 2 kompatiblen Transponder weisen eine Kapazität von mindestens 224 Bit auf, wobei ein Teil davon für Daten zur Fehlerkorrektur zur Verfügung steht.
Weiterhin existiert eine Vielzahl von Varianten mit beschreibbarem Speicher von großer Kapazität (EEPROM oder SRAM), deren Kosten entsprechend höher ausfallen.
Die mit größerem Speicher und Wiederbeschreibbarkeit gewonnene Flexibilität wird im Einzelhandel jedoch im Allgemeinen nicht benötigt. Die meisten an Einzelprodukten befestigten Tags werden nach Verwendung des Produktes vom Kunden weggeworfen. Aus Kostengründen bietet sich somit die Verwendung einfacher Speicher mit einer für Identifikationszwecke ausreichenden Größe an.

4. Einsatz von RFID im Einzelhandel

4.1 Umfrage bei Einzelhandelsunternehmen

Im Zuge der Erstellung der vorliegenden Arbeit wurden 19 deutsche Einzelhandelsunternehmen telefonisch zu ihrer Erfahrung und den Planungen zur RFID-Technologie befragt.
Ziel der Befragung war zu erfahren, mit welchen Szenarien sich die Unternehmen beschäftigen und welche positiven oder negativen Erfahrungen sie mit der jeweils eingesetzten Technologie gemacht haben. Ein interessanter Leistungswert ist neben der Zuverlässigkeitsrate vor allem die Zahl der erkennbaren Tags pro Zeiteinheit. Sie gibt Aufschluss über die Möglichkeit der Kennzeichnung von Einzelprodukten, die ja in einem Supermarktregal oft in großer Zahl auf engstem Raum platziert sind.
Eine Klassifikation der Unternehmen hinsichtlich ihrer Aktivitäten in Bezug auf die RFID-Technologie ist in Tabelle 3 aufgeführt.
Elf Unternehmen erklärten sich nicht zu Auskünften bereit, z.B. aufgrund der Firmenpolitik, wonach Anfragen dieser Art generell nicht beantwortet werden. Durch Sichtung von öffentlich zugänglichem Material wurde jedoch deutlich, dass mindestens drei der zwölf Unternehmen zumindest indirekten Kontakt mit RFID haben: C&A und Karstadt sind Mitglieder in der Organisation EPCglobal. Diese „entwickelt Standards für die einheitliche Nutzung der Radiofrequenztechnologie für Identifikationszwecke (RFID) entlang der gesamten Versorgungskette über Länder- und Branchengrenzen hinweg“ [gs1-epc]. Tchibo nutzt ein von der Gesellschaft BLG Logistics

aktiv	**zeitweise aktiv**	**beobachtend**	**nicht aktiv**	**keine Auskunft**
Metro Rewe	Kaiser's Tengelmann	Douglas Edeka Rossmann	Deichmann Schlecker	Aldi Nord Aldi Süd C&A Karstadt Kaufland Lidl Netto Norma Peek & Cloppenburg D Peek & Cloppenburg HH Tchibo

Tabelle 3: Klassifikation der 19 befragten Unternehmen hinsichtlich ihrer Aktivitäten in Bezug auf die RFID-Technologie (Quelle: eigene Erhebung)

betriebenes Hochregallager in Bremen, in dem ein RFID-System mit aktiven Transpondern die Warenbewegungen unterstützt [ziegler].
Drei der befragten Unternehmen (Douglas, Edeka, Rossmann) beobachten die Entwicklungen rund um RFID, sind aber von der Leistungsfähigkeit und der Wirtschaftlichkeit noch nicht in dem Maße überzeugt, dass sie auf einen baldigen Einsatz der Technologie abzielende Projekte durchgeführt haben. Weitere zwei Unternehmen (Deichmann und Schlecker) beschäftigen sich nach eigenen Angaben nicht mit der Technologie.
Die verbleibenden drei Unternehmen (Metro, Rewe und Kaiser's Tengelmann) berichteten von konkreten Projekten zum Einsatz von RFID, deren wichtigste Daten in Tabelle 4 aufgelistet sind.
Alle Unternehmen verwenden RFID auf den höheren Kennzeichnungsebenen und bezwecken damit vor allem die Verbesserung der Logistikprozesse. Die Metro setzt außerdem das Item Level Tagging im Future Store in Rheinberg ein, einem Verbrauchermarkt, in dem neben RFID auch andere moderne Technologien zur Demonstration und zu Tests unter Praxisbedingungen verwendet werden [metro-futurestore]. Auf diese Weise werden auch Verkaufsraum-Szenarien bearbeitet.
Alle Unternehmen bevorzugen den UHF-Frequenzbereich und das EPC-Datenformat. Das Protokoll EPCglobal Class 1 Generation 2 (kurz „Gen 2") wird von Metro und Rewe eingesetzt. Rewe begründet die Wahl von Gen 2 explizit damit, dass andere Standards nicht in gleichem Maße etabliert und propagiert sind [sandlöhken]. Kaiser's Tengelmann nutzte im einige Zeit zurück liegenden Projekt den Vorgänger Gen 1.
Die Zuverlässigkeitswerte bei Metro und Rewe liegen knapp unter der 100%-Marke, während bei Kaiser's Tengelmann in Abhängigkeit des Produktes eine große Bandbreite an Zuverlässigkeitswerten beobachtet wurde. Genauere Angaben können Tabelle 5 entnommen werden.
Aus dieser Übersicht geht auch der Einfluss von Metall und Flüssigkeit auf die Zuverlässigkeit hervor. Bei Windeln und Toilettenpapier (kein Metall, keine Flüssigkeit) gibt es keine Einschränkungen bei der Zuverlässigkeit. Bei Pizzatomaten und Backofenreiniger leidet die Erkennungsrate unter beiden Faktoren, die beobachteten Werte

	Metro	**Rewe**	**Kaiser's Tengelmann**
Zeitraum	seit 2004	seit 2003	2004/05
Kennzeichnungs-ebene	Paletten, Kartons Item Level im Future Store	größere logistische Einheiten	Paletten, Packstücke
Szenarien	alle in Kapitel 2 genannten	Wareneingang, Einlagerung, Nachschubsteuerung	Sendungsverfolgung
verwendetes Protokoll	EPC Gen 2	EPC Gen 2	EPC Gen 1
Frequenzbereich	UHF	UHF	UHF
Datenformat	EPC	EPC	EPC
Zuverlässigkeit	95-100%	98-99%	25-100%
gewünschte Leserate	keine Angabe	bis ca. 20 Tags/s	144 Tags/s
weiterer Einsatz	ja	ja	nein
Verbesserungs-bedarf	Umgang mit metallischen Umgebungen	Miniaturisierung der Transponder	Zuverlässigkeit, Kosten

Tabelle 4: Übersicht der aktiven Anwender von RFID [metro, metro-rfid, sandlöhken, kunz]

Produkt	**Maximale Erkennungsrate**
Windeln	100%
Toilettenpapier	100%
Backofenreiniger	90%
Sekt	90%
Shampoo	75%
Tomatensuppe	65%
Milch	42%
Pizzatomaten	33%
Kaffee	25%

Tabelle 5: Erkennungsraten bei Zuverlässigkeitstest von Kaiser’s Tengelmann mit dem Protokoll EPCglobal Class 1 Generation 1 [kunz]

unterscheiden sich aber sehr stark. Ein fester Abschlag auf die Erkennungsrate aufgrund von Metall und Flüssigkeit kann also nicht festgestellt werden. Das Unternehmen geht bei Verwendung des Protokolls Gen 2 zwar von deutlich besseren Zuverlässigkeitswerten aus, diese würden aber noch immer nicht durchgängig im gewünschten Bereich von mindestens 95% liegen [kunz].

Aufgrund der Kennzeichnung von größeren logistischen Einheiten stellt Rewe nur relativ geringe Ansprüche an die Leserate. Kaiser's Tengelmann hingegen möchte auch eine dreistellige Zahl an Packstücken lesen können.
Wegen der schlechten Erkennungsraten im Vergleich zur aktuell eingesetzten Barcode-Technologie (98-99%) rechnet [kunz] nicht mit einer Einführung im offenen Güterkreislauf von Kaiser's Tengelmann in den nächsten zehn Jahren. Ein weiteres Argument sind die von [kunz] geschätzten Kosten: Während für das Scannen von Barcodes an Warentoren ein PC mit Handscanner für ca. 2500 € benötigt wird, wären beim RFID-Einsatz an jedem der zahlreichen Tore in Distributionszentren und Filialen Gates zu einem Preis von grob geschätzt 15000 € nötig. Hinzu käme eine Verschlechterung der Mobilität durch die stationären RFID-Gates. Außerdem müssten momentan Dienstleister beauftragt werden, um RFID-Tags an den Produkten zu befestigen, da die Technologie nur von wenigen Produktherstellern verwendet wird.
Bei Metro und Rewe hingegen wird die Technik in Zukunft noch umfassender eingesetzt, nachdem beispielsweise bei der Metro die Prozesse am Warenein- und -ausgang „erheblich beschleunigt werden" konnten [metro-rfid]. Trotzdem sieht man auch hier in einigen Bereichen Verbesserungsbedarf.
Zu ausländischen Unternehmen, die die Technologie verwenden, zählen Wal-Mart (Gen 2 [tecchannel]), Tesco, Carrefour, Ahold (jeweils Gen 2 [flach, S. 24]), Best Buy, Marks & Spencer (jeweils Gen 2 [lxe]), Gap, Target, Albertsons, CVS/pharmacy, House of Fraser, Delhaize, Sainsbury's und Hannaford [sandlöhken-präs].

4.2 Einschätzungen weiterer Institutionen

In den Tests des Fraunhofer-Instituts für Materialfluss und Logistik (vgl. [lammers]) stellte sich heraus, dass auch bei Gen 2 nur in Ausnahmefällen Zuverlässigkeitswerte von 100% erreicht werden. Bei großen Transponderzahlen und einem schwierigen Umfeld (metallhaltige Verpackungen, Flüssigkeiten, nicht an der Oberfläche angebrachte Transponder) können die Erkennungsraten sehr stark fallen.
Bei Untersuchungen des Instituts zur Zahl der bei einer Pulklesung erkennbaren Tags hat sich eine „magische Schwelle" von 100 ergeben, wobei hier Hinweise bestehen, dass sich diese Grenze in Richtung 200 verschoben hat. Diese Zahlen wurden jedoch nur unter guten Rahmenbedingungen erreicht. Des Weiteren wurde eine recht große Abhängigkeit der erreichten Werte von der Auswahl des Lesegerätes festgestellt. Die in einigen Dokumenten [z.B. alien] erwähnten Leseraten von mehreren hundert Tags pro Sekunde konnten nicht nachvollzogen werden.
Gründe für die Verschlechterung der Leseergebnisse bei großen Transponderzahlen können einerseits in einer Überforderung des Antikollisionsverfahrens des Lesegerätes liegen. Andererseits kann aber auch eine Schwächung des vom Lesegerät erzeugten Feldes durch Transponder oder Umgebungsbedingungen wie Flüssigkeiten oder Metalle die Verschlechterung der Werte herbeiführen. Beide Ursachen lassen sich in Tests nur schwer voneinander trennen [lammers].
Der Einsatz der Technologie auf Einzelproduktebene wird von [lammers] momentan aus Kosten- und Leistungsgründen als nicht sinnvoll betrachtet. Die Verwendung auf

Kartonebene ist nach Ansicht des Instituts in Einzelfällen zu empfehlen, während die Kennzeichnung von Paletten als „auf alle Fälle sinnvoll“ eingeschätzt wird.
Wie bereits aus den oben aufgeführten Beispielen hervorgeht, scheint eine Tendenz zum Einsatz von UHF-Technologie zu bestehen, besonders das Protokoll Gen 2 erfreut sich großer Beliebtheit. So geht GS1 Germany (vgl. [füßler]) davon aus, dass, wenn Einzelhandelsunternehmen den Einsatz von RFID in die Wege leiten, dies auf Basis von Gen 2 geschehen wird, soweit es unternehmensübergreifende Anwendungen betrifft. Auch die Textilbranche – ursprünglich stark am HF-Bereich interessiert – schwenkt nach Ansicht von [füßler] zunehmend zu UHF um, während die Frequenzfrage im Pharmasektor noch nicht geklärt ist. Anlass für die baldige Einführung eines gegenüber Gen 2 verbesserten Protokolls besteht nach seiner Ansicht zurzeit nicht. Zunächst müsse Gen 2 praktisch umgesetzt werden.
Beide Institutionen und Rewe haben auch Angaben zu den vom Einzelhandel gewünschten Reichweiten gemacht. Wie aus Tabelle 6 hervorgeht und bereits in Kapitel 2 erläutert wurde, sind besonders große Werte nicht gefordert.

Rewe	**Fraunhofer**	**GS1**
1-1,5m	> 1m	ca. 2m

Tabelle 6: Angaben zu vom Einzelhandel gewünschten Reichweiten von RFID-Systemen [sandlöhken, lammers, füßler]

5. Möglichkeiten zur Leistungssteigerung

5.1 Frequenz- und Protokollauswahl

Wie bereits in Kapitel 2 ausgeführt wurde, hat die Betriebsfrequenz großen Einfluss auf die Leistungsfähigkeit eines RFID-Systems. Die Auswirkungen der Frequenz- und Protokollauswahl werden nun ausführlicher betrachtet.
Interessant sind hier vor allem Protokolle für die Luftschnittstelle zwischen Lesegerät und Transponder, von denen einige in der Norm ISO 18000 definiert sind [vgl. finkenzeller, S. 302]. Darin ist für jeden gebräuchlichen Frequenzbereich eine eigene Teilnorm enthalten (Parts 1-7).
Im Folgenden wird ein auf den HF- und UHF-Bereich beschränkter Vergleich der Leistungsdaten durchgeführt. Interessant sind somit die Normen ISO 18000-3 für den HF-Bereich und ISO 18000-6 für den UHF-Bereich. Innerhalb beider Normen sind nochmals Unternormen (Modes bzw. Types) definiert.
Neben der ISO hat sich auch die Organisation EPCglobal das Ziel der Standardisierung im RFID-Bereich gesetzt. Neben dem EPC-Datenformat (siehe Abschnitt 5.3) gehört zu den von EPCglobal definierten Standards das Protokoll EPCglobal Class 1 Generation 2 (Gen 2), während EPCglobal Class 1 Generation 1 (Gen 1) laut [füßler] nur als Vorläuferversion und nicht als Standard betrachtet wird. Das Protokoll Gen 2 wurde nach seiner Auskunft von der ISO als Norm ISO 18000-6C übernommen.

Einige Standards werden nun anhand wichtiger Leistungsmerkmale verglichen: Die HF-Protokolle ISO 18000-3 Mode 1 und 2, sowie das UHF-Protokoll Gen 2. Das Ergebnis ist in Tabelle 7 zu sehen.

	ISO 18000-3 Mode 1	**ISO 18000-3 Mode 2**	**ISO 18000-6C (Gen 2)**
Frequenz	13,56 MHz	13,56 MHz	860-960 MHz
Datenrate Kommando	26,48 kbits/s	424 kbits/s	26,7 bis 128 kbits/s
Datenrate Antwort	26,69 kbits/s	106 kbits/s pro Kanal	40 bis 640 kbits/s
Zahl der Antwortkanäle	1	8	1
Reichweite	bis 1,5 Meter	bis 1,5 Meter	bis 10 Meter
Leserate	theoretisch bis zu 40/s	bis zu 800/s	bis zu 450/s (Europa) / 880/s (USA)
Zuverlässigkeit bei gestapelten Tags	schlecht	gut	schlecht, wenn keine Zwischenräume vorhanden
Beeinflussung der Zuverlässigkeit und Reichweite durch Metall / Flüssigkeiten	relativ klein	relativ klein	relativ groß
Lesbarkeit sich bewegender Tags	keine Angabe	gut	beschränkt

Tabelle 7: Vergleich von HF- und UHF-Standards [magellan, alien]

Es wird deutlich, dass das Protokoll ISO 18000-3 Mode 1 dem Mode 2-Protokoll in allen Bereichen entweder unterlegen ist oder zumindest keine besseren Leistungsmerkmale aufweist.
Der Vergleich zwischen den beiden verbleibenden Protokollen ist weniger eindeutig. Während das HF-Protokoll bei der Datenrate, der Zuverlässigkeit und der Anpassung an in der Praxis relevanten Umgebungsbedingungen überlegen ist, hat das UHF-Protokoll bei der Reichweite Vorteile. Bei der für das Item Level Tagging so wichtigen Leserate ist ein eindeutiges Urteil wegen der national unterschiedlichen Frequenzbänder nicht möglich.
Wie aus den Befragungen der Einzelhandelsunternehmen hervorgegangen ist, hat gerade die Reichweite der Technologie für die Unternehmen eine eher geringe Bedeutung, teilweise wird eine kleine Reichweite sogar bevorzugt. Sie kann zum Teil auch vom HF-Protokoll erreicht werden. Probleme mit UHF-Protokollen wurden von einigen Befragten in der Zuverlässigkeit gesehen. Hier zeigt der Vergleich der Leistungsmerkmale bessere Voraussetzungen beim HF-Protokoll. Auch bei der Leserate ist es auf Basis der europäischen Frequenzregulierung überlegen.
Somit kann anhand dieser Daten zumindest in Europa eher zu einer Verwendung des ISO 18000-3 Mode 2 geraten werden, falls eine Reichweite von 1,5 Metern als ausreichend erachtet wird. Dies steht im Widerspruch zum breiten Einsatz des Gen 2-Protokolls, der aus der Befragung der Einzelhandelsunternehmen hervorgegangen ist.

Inzwischen hat sich EPCglobal entschieden, einen dem Gen 2 ähnlichen Standard auch im HF-Segment einzuführen, wobei dieses Protokoll jedoch ausdrücklich als für die Warenlogistik weniger geeignet betrachtet wird [gs1-hf].
An dieser Stelle sei noch einmal an die Aussage von [lammers] erinnert, der die in Tabelle 7 angegebenen Werte für die Leserate des Gen 2-Protokolls in Tests nicht nachvollziehen konnte. Die angegebenen Werte für das HF-Protokoll Mode 2 entstammen von einem Unternehmen, das dieses Protokoll vermarktet. Dem Verfasser dieser Arbeit liegen keine Erfahrungswerte einer unabhängigen Stelle vor, so dass die Leistungswerte evtl. mit Vorsicht zu betrachten sind. Auch [bitkom, S. 26] verweist darauf, dass in Veröffentlichungen angegebene Messwerte auch durch idealisierte Rahmenbedingungen herbeigeführt werden können, da keine standardisierten Tests definiert sind.

5.2 Zugriffsverfahren[2]

Wesentlichen Einfluss auf die Leistungsfähigkeit eines Standards im Hinblick auf die Zahl der erkennbaren Tags pro Zeiteinheit haben Vielfachzugriffs- und Antikollisionsverfahren. Wenn sich eine große Zahl von Transpondern zur gleichen Zeit im Ansprechbereich eines Lesegerätes befindet, muss das Lesegerät in der Lage sein, die Kapazität der vorhandenen Kommunikationskanäle den Transpondern so zuzuweisen, dass keine gegenseitige Störung durch Überlagerung der Signale auftritt.
Besonders Zeitmultiplexverfahren sind bei RFID-Systemen gebräuchlich. Hierbei nutzen die Transponder zeitlich versetzt die verfügbare Kanalkapazität. Prinzipiell kann die Steuerung der Methode vom Lesegerät oder vom Transponder vorgenommen werden. Im letzten Fall treten jedoch Probleme mit Geschwindigkeit und Flexibilität auf, so dass meist das Lesegerät die Verfahrensregelung übernimmt. Ein Beispiel für die Transpondersteuerung ist das probabilistische ALOHA-Verfahren. Bei der Steuerung durch das Lesegerät wird zwischen Polling- und Binary-Search-Verfahren unterschieden. Beim Polling muss dem Lesegerät eine Liste der möglichen Transponderkennungen vorliegen, die der Reihe nach abgearbeitet wird.
Das Binary-Search-Verfahren oder Tree-Walking-Verfahren wird im Folgenden exemplarisch vorgestellt. Es wird beispielsweise im Protokoll ISO 18000-6B verwendet [hightechaid]. Hierbei wird absichtlich eine Kollision mehrerer zeitgleich gesendeter Datenpakete erreicht, wobei erkannt werden kann, an welchen Bitpositionen Kollisionen auftreten.
Dies ist nur bei bestimmten Codierungsformen wie der Manchester-Codierung möglich (siehe Abbildung 2).
Bei der Manchester-Codierung wird eine logische Eins durch eine fallende Flanke des Pegels angezeigt, eine logische Null durch eine steigende. Wenn kein Pegelwechsel stattfindet, liegt ein Fehler vor. Bei Überlagerung von zwei gesendeten Einsen kann das Lesegerät so erkennen, dass sämtliche Transponder an der jeweiligen Bitposition eine Eins gesendet haben. Werden verschiedene Wertigkeiten übertragen, registriert das Lesegerät einen Fehler in Form einer bitweisen Kollision.

[2] [finkenzeller, Kapitel 7.2]

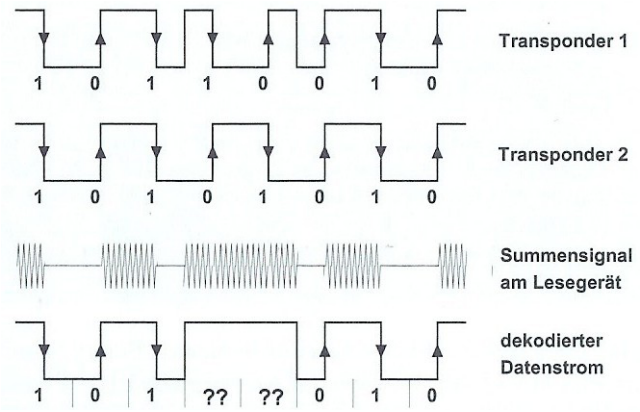


Abbildung 2: Erkennen von Kollisionen im Datenstrom bei Manchester Codierung (entnommen aus [finkenzeller])

Bei anderen Codierungsformen, bei denen z.B. eine logische Eins durch einen hohen Pegel und eine logische Null durch einen niedrigen Pegel signalisiert wird, kann das Lesegerät nicht auseinander halten, ob alle Transponder eine Eins senden oder nur ein Teil der Transponder.

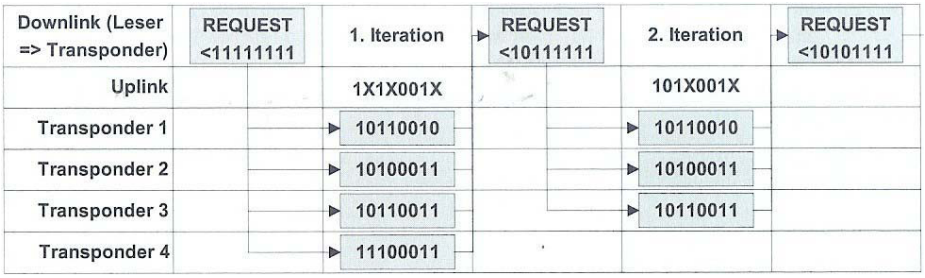


Abbildung 3: Ablauf des Binary-Search-Verfahrens (entnommen aus [finkenzeller])

Beim Binary-Search-Verfahren (Abbildung 3) werden vom Lesegerät iterativ Abfragen mit einem Request-Kommando ausgeführt. Auf die erste Anfrage REQUEST(<= 11111111) antworten sämtliche Transponder, da als Parameter die höchstmögliche Identifikationsnummer gewählt wurde. Anschließend erfolgt eine Selektion an der höchstwertigen Bitposition, an der eine Kollision festgestellt wurde, indem im Parameter des Request-Kommandos an dieser Position eine Null gesetzt wird. Die Zahl der aufgerufenen Transponder wird somit sukzessiv verringert, bis nur noch ein Tag auf die Anfrage antwortet. Seine Identifikationsnummer kann nun kollisionsfrei festgestellt werden. Anschließend beginnt das Verfahren von vorne, wobei bereits registrierte Transponder nicht mehr auf neuerliche Anfragen antworten. Es wird deutlich, dass bei diesem Verfahren nur das Lesegerät einen Überblick über die empfangenen Daten erhält, so dass hier keine Transpondersteuerung in Frage kommt.

Beim Binary-Search-Verfahren existieren dynamische Erweiterungen, bei denen die Zahl der zu übertragenden Bits verringert wird. Beispielsweise reicht in Abbildung 3 bei der zweiten Iteration im Request-Kommando bereits die Angabe der ersten zwei Stellen („10“) aus, um die drei verbleibenden Transponder anzusprechen. In der Antwort der Transponder kann hingegen auf das Senden dieser zwei Stellen verzichtet werden, da sie dem Lesegerät bereits bekannt sind. Somit kann die Zahl der zu übertragenden Bits halbiert werden, so dass die Geschwindigkeit des Verfahrens in etwa verdoppelt wird.
Weiterentwicklungen der Zugriffsverfahren sind somit ein weiterer Weg zur Steigerung des Leistungsniveaus von RFID-Systemen.

5.3 Datenformat EPC[3]

Auch das verwendete Datenformat kann für die Skalierbarkeit eines RFID-Systems von Bedeutung sein. Der Electronic Product Code (EPC), der von allen in der Umfrage von Kapitel 4 identifizierten Anwendern der RFID-Technologie verwendet wurde, bietet hierzu einige Möglichkeiten.
Für den EPC existieren diverse Codierungsschemata mit einem Umfang von 64 oder 96 Bit. In Tabelle 8 ist exemplarisch das Format SGTIN-96 (serialized global trade item number mit 96 Bit) beschrieben, das von der Metro Gruppe zur Kennzeichnung von Kartons und Artikeln verwendet wird [metro-leitlinien].

Kopf	Filterwert	Partition	Firmen-nummer	Artikel-nummer	Serien-nummer
8 Bit	3 Bit	3 Bit	20-40 Bit	4-24 Bit	38 Bit

Tabelle 8: Zusammensetzung der EPC-Kodierung SGTIN-96 [vgl. finkenzeller, metro-leitlinien]

Interessant für die Möglichkeiten zur Leistungssteigerung ist vor allem der Filterwert. Mit seiner Hilfe können verschiedene Kennzeichnungsebenen selektiv angesprochen werden [vgl. epc-tag, S. 27], z.B. kann sich eine Anfrage des Lesegerätes nur an alle auf Kartons befestigten Tags richten. Wenn sich alle nicht angesprochenen Tags, die sich bei einer Filterung im Ansprechbereich des Lesegerätes befinden, passiv verhalten, werden weniger Daten gleichzeitig an das Lesegerät gesendet, so dass das Zugriffsverfahren schneller ablaufen kann. Im Prinzip lassen sich auch die weiteren Datenfelder, z.B. Firmen- und Artikelnummer für eine solche Filterung nutzen.

[3] [finkenzeller]

6. Fazit

Ein verstärkter Einsatz von RFID-Technologien im Einzelhandel zeichnet sich für die kommenden Jahre ab. Die Gründe dafür liegen in den Szenarien, die mit Hilfe von RFID umgesetzt oder erleichtert werden können. Viele dieser Szenarien, insbesondere diejenigen, die eine Kennzeichnung auf Einzelproduktebene voraussetzen, hängen jedoch von der Leistungsfähigkeit der Technologie ab.
In dieser Arbeit wurde deutlich, dass die Technologie für ein umfassendes Item Level Tagging wegen zu hoher Kosten und zu geringer Skalierbarkeit noch nicht reif ist. Szenarien, die nur eine Kennzeichnung größerer logistischer Einheiten voraussetzen, sind jedoch bereits umsetzbar.
Im Einzelhandel scheint sich das Protokoll EPCglobal Class 1 Generation 2 immer mehr durchzusetzen. Die Informationen von Rewe zeigen, dass eine Entscheidung für Gen 2 nicht nur aus reinen Leistungsgesichtspunkten getroffen wird, sondern auch wegen der Verbreitung des Standards bei Herstellern und der Konkurrenz. Allerdings besteht nach Ansicht einiger Unternehmen noch Verbesserungsbedarf bei der Zuverlässigkeit der Technologie, dem Umgang mit Metallen und der Leserate. Auch eine weiter gehende Miniaturisierung der Transponder wird gewünscht. Wie ein Vergleich mit dem Protokoll ISO 18000-3 Mode 2 gezeigt hat, ist dieses bei Betrachtung der Leistungsdaten sogar überlegen, vorausgesetzt, es wird keine Reichweite über 1,5 Metern benötigt.
Die Wege zu einer leistungsfähigeren Technologie führen über die Entwicklung neuer Protokolle, dem Einsatz besserer Zugriffsverfahren sowie der Ausnutzung der durch das Datenformat gebotenen Möglichkeiten zur Selektion von Transpondergruppen.

Quellenverzeichnis

[alien] Whitepaper “EPCglobal Class 1 Gen 2 RFID Specification”, Alien Technology, Morgan Hill (USA), http://www.alientechnology.com/docs/AT_wp_EPCGlobal_WEB.pdf, Version vom 16.01.2007

[bitkom] Bitkom RFID Guide 2006, Bundesverband Informationswirtschaft, Telekommunikation und neue Medien e.V., Berlin, http://www.bitkom.org/files/documents/rfid_guide_2006.10.11_ST.pdf, Version vom 7.01.2007

[epc-tag] EPC Generation 1 Tag Data Standards Version 1.1 Rev. 1.27, EPCglobal, Mai 2005, http://www.epcglobalinc.org/standards/EPCglobal_Tag_Data_Standard_TDS_Version_1.1_Revision_1.27.pdf, Version vom 4.02.2007

[finkenzeller] Finkenzeller, K., RFID-Handbuch, 4. Auflage, 2006, Carl Hanser Verlage, München

[flach] Flach, A. B., Doing Business in the era of RFID, Amsterdam, März 2006, http://www.rfidjournal.com/whitepapers/download/175 (Anmeldung erforderlich), Version vom 8.01.2007

[füßler] Füßler, A., Leiter Forschung und Entwicklung RFID, GS1 Germany, Köln, Telefoninterview mit dem Verfasser dieser Arbeit am 15.01.2007

[gs1-aut] Vor- und Nachteile der Radiofrequenzidentifikation, GS1 Austria, Wien, http://www.gs1austria.at/epc/html/11_1_5tech.html, Version vom 16.01.2007

[gs1-epc] EPCglobal, GS1 Germany, Köln, http://www.gs1-germany.de/internet/content/produkte/epcglobal/index_ger.html, Version vom 20.01.2007

[gs1-faq] RFID-Infocenter: Beschreiben und Lesen von Transpondern, GS1 Germany, Köln, http://www.gs1-germany.de/internet/content/produkte/epcglobal/rfid_epc_in_der_praxis/rfid_infoboerse/lesbarkeit_von_tags/index_ger.html, Version vom 16.01.2007

[gs1-hf] EPCglobal ergänzt RFID-Standards um HF-Luftschnittstelle, Pressemeldung, GS1 Germany, Köln, http://www.gs1-germany.de/VHM/internet/internet/content/presse/index_ger.html?nid=news_items2471&lang=ger, Version vom 28.01.2007

[hightechaid] ISO/IEC 18000 Interface Standards, High tech aid, Pittsburgh (USA), http://www.hightechaid.com/standards/18000.htm, Version vom 28.01.2007

[kunz] Kunz, U., Logistikleitung, Kaiser`s Tengelmann AG, Viersen, Telefoninterview mit dem Verfasser dieser Arbeit am 11.01.2007

[lammers] Lammers, W., Fraunhofer-Institut für Materialfluss und Logistik, Dortmund, Telefoninterview mit dem Verfasser dieser Arbeit am 17.01.2007

[lxe] RFID Technology for Warehouse & Distribution Operations, LXE Inc., Norcross (USA), Juni 2006, http://www.lxe.com/pdf/White_Paper-RFID_in_the_WH_and_DC.pdf, Version vom 28.01.2007

[magellan] Whitepaper "A comparison of RFID frequencies and protocols", Magellan Technology, Annandale (Australien), März 2006, http://www.magtech.com.au/folder.2006-01-16.8737780017/downloads/white-paper-frequency-comparison-31-march-2006-1.pdf, Version vom 8.01.2007

[metro] Metro Group RFID Competence Team, E-Mail an den Verfasser dieser Arbeit am 19.01.2007

[metro-frequenz] RFID-Funktionsweise, Metro Group, Düsseldorf, http://www.future-store.org/servlet/PB/menu/1007866_l1_yno/index.html, Version vom 16.01.2007

[metro-futurestore] Broschüre "Willkommen in Future Store", Metro Group, Düsseldorf, August 2006

[metro-ic] Broschüre "RFID Innovation Center", Metro Group, Düsseldorf, Oktober 2006

[metro-leitlinien] Broschüre "Leitlinien für die Einführung von RFID bei der METRO Group", Metro Group, Düsseldorf, November 2006

[metro-rfid] RFID@Metro, Metro Group, Düsseldorf, http://www.future-store.org/servlet/PB/ menu/1007240_l1_yno/index.html, Version vom 23.01.2007

[sandlöhken] Sandlöhken, J., Projektleitung RFID, Rewe Gruppe, Köln, Telefoninterview mit dem Verfasser dieser Arbeit am 26.01.2007

[sandlöhken-präs] Sandlöhken, J., Handelsaspekte beim Thema RFID aus Sicht der Rewe Gruppe, Rewe Gruppe, Köln, http://www.duesseldorf.ihk.de/de/Anlagen/Sonstiges/V5_Funketikett_231105_Sandloehken.pdf, Version vom 17.01.2007

[tecchannel] Wal-Mart keeps RFID momentum after personnel reshuffle, IDG Business Media, München, http://www.tecchannel.de/news/international/437232/, Version vom 28.01.2007

[wikipedia-rfid] Radio Frequency Identification, Wikipedia Foundation, St. Petersburg (USA), http://de.wikipedia.org/wiki/Rfid, Version vom 18.01.2007

[wikipedia-transponder] Transponder, Wikipedia Foundation, St. Petersburg (USA), http://de.wikipedia.org/wiki/Transponder, Version vom 16.01.2007

[ziegler] Ziegler, A., RFID im Lagerwesen - Praxisbeispiel Tchibo Non-Food-Lager Bremen, Manager Operation, BLG Logistics, Bremen, http://www.ihk-nordwestfalen.de/netzwerk_logistik/bindata/Vortrag_Ziegler_22092004.pdf, Version vom 24.01.2007

Discovery und Querying in P2P-Netzen

Thomas Beck

Universität Karlsruhe (TH) email@th-beck.de

1 Einleitung

Eine der Visionen, wie sich das Internet in Zukunft entwickeln könnte ist das Internet der Dinge. Jedes Gegenstand kann dabei Anbindung an das Internet haben und jeder Nutzer hat über das Internet Zugriff auf alle Dienste und Geräte. Durch Mobilität wird es zudem zunehmend komplexer, benötigte Informationen und Dienste in dieser stark expandierenden Umgebung zu finden. Neben der reinen Information ist Lokation einer Information von immer zentralerer Bedeutung. Daher ist das Hauptziel in großen Peer-to-Peer-Anwendungen effiziente Algorithmen zur Verfügung zu stellen, die neben Informationssuche auch Lokation und Routing unterstützen. In dieser Seminararbeit werden unterschiedliche Peer-to-Peer Architekturen anhand von zwei praxisnahen Beispielen miteinander verglichen. Neben einer verteilten Hash-Funktion, implementiert in INS/Twine[1], und einem virtuellen Overlaynetzwerk mit voneinander unabhängigen Peergroups, implementiert in JXTA-C[2], werden noch zwei weitere Ansätze betrachtet. Durch SLP[3] wird untersucht, inwiefern ein bereits verbreiteter Standard die neuen Aufgaben lösen kann. Mit Jini[4] wird abschließend ein Ansatz betrachtet, der mit Hilfe von Service-Objekten in Java versucht diese Aufgaben effizient zu erfüllen. Um die Ansätze miteinander vergleichbar zu machen und die Anforderungen die an ein solches System gestellt werden besser zu verdeutlichen, folgen zwei Beispiele auf die im Laufe des Aufsatzes wiederholt Bezug genommen wird.

1.1 Beispiel 1 (Paketverfolgung)

Ein Logistikunternehmen plant eine Nachverfolgung von Paketen. Dabei soll ein Mitarbeiter oder Kunde die Möglichkeit haben über das Internet eine Historie über die verschiedenen Verladungsabschnitte abrufen zu können. Ein möglicher Ablauf der Paketverladung kann dabei aus beliebig vielen Abschnitten bestehen und ist nicht im Voraus bekannt. Sobald ein Kunde ein Paket in der Filiale abgibt scannt, ein Angestellter eine Paket-ID und leitet das Paket weiter. Über mehrere Umladestationen wird das Paket mit verschiedenen Verkehrsmitteln transportiert und weiterverladen. Am Zielort angekommen, wird es einem Auslieferungsfahrzeug zugeteilt und schließlich ausgeliefert. Zusammengefasst enthält dieses Anwendungsbeispiel folgende Eigenschaften und Anforderungen:

Anwendungseigenschaften:

- Paket durchläuft viele verschiedene Netze (z.B. im Transporter, in jeder Halle, in LKW / Flugzeug)
- Lokalitätsinformation wichtig (Kunde: „Wo ist mein Paket jetzt?“)
- Starke Mobilität aller beteiligten Objekte
- Beschränkte Flexibilität der Anforderungen, Anwendung ist im Voraus bekannt
- Netztopologie bekannt

Typische Anfragen:

1. Netzintern
   - Welche Menge an Paketen befindet sich in Netz X? (Logistikorganisation)
2. Netzübergreifend
   - Historie (Wo war Paket X Schritt für Schritt?)
   - Lokalität (Wo befindet sich Paket X gerade jetzt?)
   - Statistiken (Welche Anzahl an Paketen wurde dieses Jahr transportiert?)

Anforderungen:

- Suchen verteilter Information
  (z.B. Historie aus mehreren Datenbanken zusammensetzen)
- Auffinden eines speziellen Objektes (vgl. Lokalität)

1.2 Beispiel 2 (*Finden lokal verfügbarer Ressourcen)*

Ein Anwender befindet sich in einer ihm unbekannten Umgebung und ist auf der Suche nach einem gewissen Dienst. Dabei kann es sich beispielsweise um ein Festnetztelefon, einen Drucker, einen Getränkeautomaten, einen Schnellimbiss, einen EC-Automaten oder einen beliebigen anderen Dienst bzw. Service handeln. Diesen Dienst kann er durch Attribute relativ präzise (abhängig vom Anwendungsfall) beschreiben, aber die Lokalität ist ihm unbekannt. Eine Suchanfrage sollte somit beantworten, wo sich der nächste für diesen Anwender erreichbare und verfügbare Dienstanbieter befindet und eine möglichst genaue Beschreibung dieses Dienstes zur Verfügung stellen. Die Eigenschaften und Anforderungen dieses Anwendungsfalles unterscheiden sich deutlich von dem ersten Beispiel. Diese sind:

Anwendungseigenschaften:

- Lokalität ist wichtigste Information
- Service-Erbringer üblicherweise nicht mobil, Nutzer ist mobil
- Nutzer befindet sich in ihm unbekannter Umgebung
  - erfordert automatische Konfiguration (z.B. über DHCP)

Typische Anfragen:

- Am „nächsten“ befindlicher Dienst mit gewünschten Eigenschaften
  - Wie komme ich dort hin?

Anforderungen:

- Wie sollen Dienste in Netzen organisiert werden?
- Sicherheit: Wie kann sichergestellt werden, dass nicht jeder diesen Dienst verwenden kann? (z.B. Drucker im Institut → nicht jeder darf von der Straße aus Dateien drucken)
- Neben Lokalität ist Erreichbarkeit sehr wichtig! (z.B. Drucker in Hochhaus nebenan; Feuerlöscher in Gebäude auf anderer Straßenseite)

Da Lokalität in vielen Anwendungsszenarien einen zentralen Stellenwert einnimmt, ist es naheliegend, die Netzwerkstruktur daran anzupassen. Eine hierarchische Struktur bietet die Möglichkeit lokal benachbarte Dienste zu gruppieren und somit leichter ausfindig zu machen. Jedoch entsteht dadurch auch die Gefahr, an Wurzelknoten einen Flaschenhals zu erzeugen.

2 Beschreibung der vorgestellten Systeme

2.1 INS/Twine

INS/Twine versucht die Nachteile, die eine hierarchische Organisation der Verzeichnissuchdienste (im Folgenden *Resolver* genannt) mit sich bringt zu umgehen. Je nach Kriterium, welches für die Organisation einer Hierarchie herangezogen wird, können Anfragen gefunden werden, die das rekursive Durchlaufen des ganzen Baumes erfordern. Häufig wird Lokalität verwendet um eine Resolverhierarchie aufzubauen, dabei verwendet beispielsweise jedes Gebäude einen eigenen Resolver. Die nächste Ebene wäre beispielsweise ein übergeordneter „Firmenresolver“, der wiederum einem „Stadtresolver“ untergeordnet ist. Anfragen, die sich auf eine spezielle begrenzte Umgebung beziehen, können mit diesem Ansatz gut beantwortet werden. Jedoch ist es aufwändig beispielsweise alle Fileserver in einer gegebenen Stadt herauszufinden. Der dadurch an der Wurzel der Hierarchie entstehende Engpass wird durch INS/Twine umgangen.

Strands INS/Twine extrahiert aus einer Ressourcenbeschreibung jede eindeutige Teilsequenz an Attributen und Attributwerten. Jede solche Teilsequenz wird *Strand* genannt.[1] Die XML-Ressourcenbeschreibung wird dazu in einem ersten Schritt als Baum dargestellt und anschließend werden ausgehend vom root-node alle Pfade zu den Blättern abgelaufen und extrahiert. Dieses Vorgehen wird in nachfolgendem Beispiel verdeutlicht[1].
Für jeden einzelnen Strand wird dann ein Hash-Wert berechnet, der verwendet wird, um den Strand auf einen Resolver abzubilden. Dieser Resolver erhält anschließend das gesamte Advertisement und kann somit Anfragen, die den gleichen Strand beinhalten, komplett bearbeiten. Abhängig vom Typ der Nachricht wird

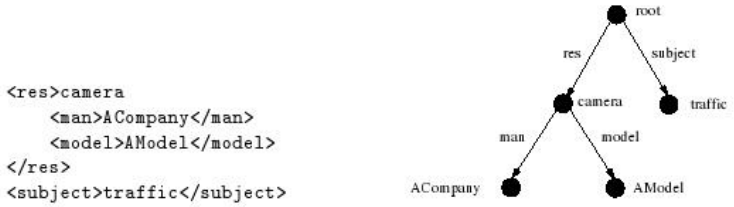


Abbildung 1. Abbilden einer XML-Ressourcenbeschreibung auf einen AV-Tree

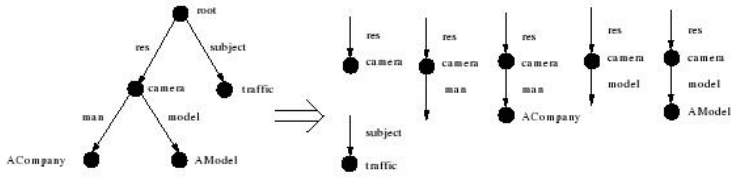


Abbildung 2. Extrahieren von Strands aus einem AV-Tree

dem so ermittelten Resolver entweder ein Advertisement oder eine Suchanfrage geschickt. Durch die Eigenschaften der Hash-Funktion werden ähnliche Strands im Allgemeinen nicht auf den gleichen Resolver abgebildet. Im obigen Beispiel könnte also jeder Strand auf einen anderen Resolver abgebildet werden.

Netzstruktur / Registrierung Das INS/Twine Netzwerk besteht aus Clients und Resolvern, wobei Resolver für Routing und Anfragebearbeitung verantwortlich sind. Die Resolver organisieren sich dabei selbst in einem Overlay-Netzwerk um Ressourcenbeschreibungen zu verbreiten und um gemeinsam Anfragen von Clients zu beantworten. Dazu ist es notwendig, dass jeder Resolver einen Teil der vorhandenen Resolver im Netzwerk kennt. Wenn ein Client einen Dienst im Netzwerk zur Verfügung stellen will, registriert er diesen bei einem Resolver zu dem er direkten Zugang hat mit Hilfe einer XML-Ressourcenbeschreibung. Kommunikation zwischen Clients und Resolvern wird dabei auch Kommunikation am *Rand des Netzwerks*, Kommunikation zwischen Resolvern im *Kern des Netzwerks* genannt.

Suche Um einen Dienst innerhalb des Netzwerks zu finden generiert ein Client eine gewünschte Ressourcenbeschreibung und sendet diese an seinen zugeordneten Resolver. Der Resolver extrahiert daraus alle Strands und sendet partielle Suchanfragen an die daraus ermittelten Resolver. Aus den erhaltenen Antworten wählt er den besten Treffer anhand des längsten Strands aus und sendet diesen an den Client zurück. Dieses Vorgehen verdeutlicht, dass sehr kurze und dadurch häufig vorkommende Strands, in der Praxis nicht verwendet werden um Suchanfragen zu beantworten. Im Normalfall unterscheidet sich das Ergebnis also nicht wenn Resolver einen Schwellwert verwenden, um zu häufige Einträge

zu verwerfen. Durch den Einsatz von Strands ist bereits sichergestellt, daß eine Suchanfrage bestmöglich beantwortet wird. Selbst wenn es keine exakte Übereinstimmung mit der gewünschten Dienstbeschreibung gibt, beantwortet der Resolver eine Anfrage mit dem am besten passenden Dienst, denn der gefundene Dienst stimmt in den wichtigsten Attributen nahe des Wurzelknotens mit der Suchanfrage überein.

2.2 Project JXTA-C

JXTA-C unterteilt im Gegensatz zu INS/Twine die sich im Netzwerk befindlichen Geräte in kleinere Gruppen. Diese Gruppen werden *Peergroups* genannt, sind charakterisiert durch eine eindeutige *PeergroupID* und größtenteils getrennt von Geräten die sich in anderen Peergroups befinden. Über das bereits bestehende physikalische Netzwerk erzeugt JXTA-C ein virtuelles Netzwerk welches eine einheitliche Addressierung und Abstraktion von den darunter liegenden Netzwerkeigenschaften ermöglicht (siehe Abbildung). Peers eines JXTA-C Netzwerkes werden eindeutige 128-bit IDs zugewiesen, die sich selbst dann nicht ändern, wenn sich das gleiche Gerät über verschiedene Schnittstellen mit dem Netzwerk verbindet.

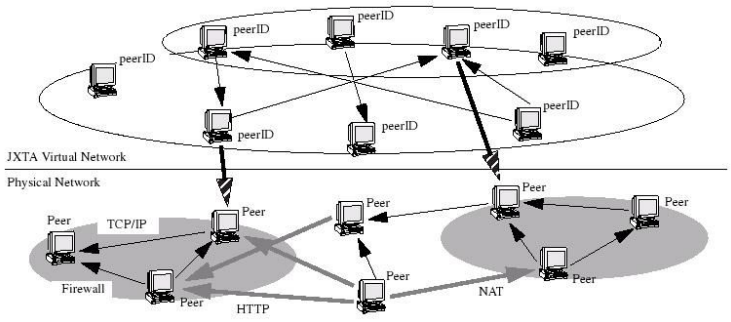


Abbildung 3. Netzstruktur in JXTA [2]

Netzstruktur Geräte, die die gleichen Interessen besitzen, formen eine eigene Peergroup. [2] Um einen Dienst zu nutzen ist es also notwendig eine Peergroup zu finden, die sich mit der Erfüllung dieses Dienstes beschäftigt. Neben einer eindeutigen PeergroupID wird jede Peergroup durch die Menge an Services, die sie verbindet, charakterisiert. Diese Menge an Services wird *peergroup services* genannt und enthält neben grundlegenden Diensten die den Betrieb sicherstellen (z.B. *discovery*, *pipe*, *rendezvous*) beliebig erweiterbare Elemente. Generell ist nicht festgelegt zu welchem Zeitpunkt oder durch welchen Anlass eine neue Peergroup gebildet wird, sondern JXTA-C stellt lediglich Mechanismen zur

Verfügung wie eine Peergroup erzeugt, bekannt gemacht und gefunden wird.[2] Dadurch ergibt sich ein Problem, wie die Gruppierung und dadurch Strukturierung des Netzes optimal erfolgt. Es liegt alleine an Anwendungsentwicklern und Netzwerkadministratoren Peergroups dynamisch zu erzeugen und zu optimieren, um den Anforderungen der Benutzer und Anwendungen gerecht zu werden. Durch die Trennung des Netzes in verschiedene Peergroups ermöglicht JXTA-C implizit die Verwendung von Wirkungsbereichen und vermindert dadurch den Overhead durch Suchanfragen.

Routing Nachrichten, die aus XML oder binär-Daten bestehen können, werden prinzipiell durch alle teilnehmenden Geräte geroutet. Es besteht jedoch die Möglichkeit das Routing durch sich freiwillig dazu bereit erklärende *Relay Peers* zu vereinfachen. Diese speichern Routing Tabellen um damit Nachrichten effizient weiterzuleiten. Meldet sich ein neuer Peer in einem Netz an, kann er in seiner Anmeldung auch einige bevorzugte Relay Peers in seiner Nähe angeben. Durch diesen Mechanismus können Nachrichten sogar für vorübergehend nicht erreichbare Peers zwischengespeichert und zeitversetzt übermittelt werden.

Suche Um eine neue Ressource in einem Netz zur Verfügung zu stellen, muss ein Peer diese Ressource durch eine standardisierte XML-Beschreibung bekannt machen. Diese Advertisements werden von anderen Peers zwischengespeichert und weiter verbreitet. Um Konsistenz innerhalb des Netzes sicherzustellen, sind alle Advertisements mit einer Lebenszeit versehen, nach der sie verfallen. Die dadurch gewonnene Robustheit erfordert allerdings eine regelmäßige Aktualisierung und somit einen etwas höheren Overhead. Zwischengespeicherte Advertisements werden von Peers verwendet, um Suchanfragen nach Ressourcen zu beantworten und um selbst verfügbare Ressourcen zu finden. Ähnlich dem Konzept der Relay Peers gibt es auch für Advertisements Geräte, die eine große Anzahl an Advertisements zwischenspeichern und somit innerhalb einer Peergroup eine bevorzugte Anlaufstelle für Suchanfragen darstellen. Diese Geräte werden *Rendezvous Peers* genannt. Durch Verdrängung der Advertisements nach dem Least Recently Used Schema (LRU) besteht unabhängig von der Größe der Peergroup eine hohe Performanz für häufig angefragte Dienste. Diese werden dann meist innerhalb eines Hops gefunden, da sich deren Advertisement mit hoher Wahrscheinlichkeit im Zwischenspeicher des Rendezvous Peers befinden.[2]
Ein spezialisierter Suchmechanismus nach Advertisements kann durch den Anwendungsentwickler für jede Peergroup individuell programmiert werden. JXTA-C implementiert diesen nur, um für den Betrieb essentiell wichtige Advertisements (beispielsweise Discovery und Rendezvous) zu verbreiten. Dadurch wird größtmögliche Flexibilität bereitgestellt, da je nach Anforderung eine zentralisierte, dezentralisierte oder hybride Strategie eingesetzt werden kann. Ein Rendezvous Peer kann Anfragen an benachbarte Rendezvous Peers weiterleiten um sie zu beantworten. Für einen normalen Peer, der nicht die Rolle eines Rendezvous Peers übernommen hat, ist es jedoch nicht gestattet Anfragen nach Advertisements weiterzuleiten um ein sonst enstehendes Fluten der Peergroup

zu unterbinden. Die Anzahl Hops, die eine Suchanfrage dabei von Rendezvous Peers weitergereicht werden kann, ist dabei individuell konfigurierbar.

2.3 SLP

Das Service Location Protol [4] ist eines der ältesten Protokolle um Ressourcen und Dienste in heterogenen Netzen zu finden. Die Architektur eines SLP-Netzes besteht aus folgenden drei Komponenten:

Tabelle 1. Komponenten von SLP

Komponente	**Funktion**
User Agent (UA)	Repräsentiert eine Anwendung, die Dienst benötigt
Service Agent (SA)	Repräsentiert einen verfügbaren Dienst und dessen Eigenschaften
Directory Agent (DA)	Verzeichnisdienst

Netzstruktur Ein *User Agent* (UA) läuft auf jedem Gerät und repräsentiert eine Anwendung wodurch der UA die Eigenschaften des gewünschten Dienstes kennt. Das entsprechende Gegenstück um einen Dienst im Netzwerk zur Verfügung zu stellen ist der *Service Agent* (SA). Er läuft auf jedem Gerät, das einen Dienst zur Verfügung stellt und kommuniziert bei der Nutzung eines Dienstes mit dem UA des anfragenden Systems. Der *Directory Agent* (DA) ist ein Verzeichnisdienst der für die Suche und Registrierung eines Services benötigt wird. Je nach Größe des Netzwerks ist der Betrieb ohne DAs möglich (nur in sehr kleinen Netzen, daher hier nicht weiter betrachtet) oder mehrere DAs befinden sich im Netzwerk. In Abbildung 4 wird das Zusammenspiel der Komponenten verdeutlicht.

Discovery Ein Gerät, das sich neu mit dem Netzwerk verbindet, muß in einem ersten Schritt einen Directory Agent im Netzwerk finden. Dazu gibt es verschiedene Möglichkeiten wie dies geschehen kann, wobei eine statische Konfiguration aller Geräte zwar möglich ist, aber in einer ubiquitären Umgebung keine Alternative darstellt. Ein UA oder SA kann aktiv durch Broadcast einer DA-Suchanfrage nach DAs suchen, worauf im Netzwerk vorhandene DAs mit einem Unicast Advertisement darauf antworten. Eine passive Möglichkeit ist es einen Broadcast eines DAs abzuwarten, wodurch dieser sich innerhalb seines Netzwerkes bekannt macht. Eine dritte Möglichkeit ist es DHCP-Option (78) zu verwenden um die notwendige Adresse des DAs zu erhalten.[5]

Dienstregistrierung Sobald ein SA einen oder mehrere DAs innerhalb seines Netzes kennt, registriert er sich bei diesen durch eine Beschreibung seines zur

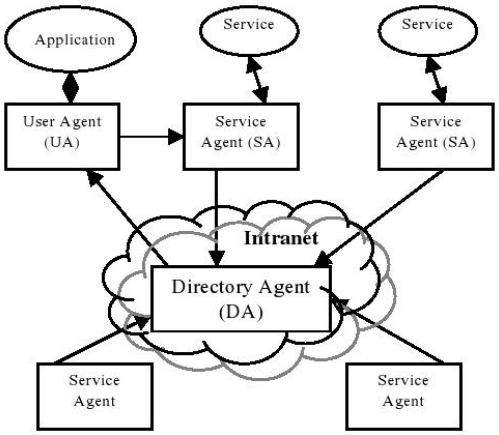
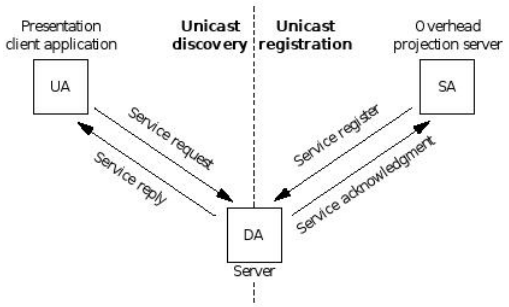


Abbildung 4. Netzstruktur in SLP [4]

Verfügung gestellten Dienstes. Ein UA kann, nachdem er einen DA gefunden hat, an diesen Suchanfragen stellen um benötigte Dienste zu finden. Das Zusammenspiel der Komponenten ist in der folgenden Abbildung [5] verdeutlicht.



Die Nachricht um einen Dienst zu registrieren besteht grundsätzlich aus zwei Teilen, einer *Service URL* und einem dahinter folgenden *Service Template*, das Attribut-Werte Paare enthält, um den Dienst genauer zu beschreiben. Mit Hilfe der Service URL kann ein UA den Dienst verwenden und anhand des Service Templates ist es möglich eine genauere Suche durchzuführen.

service:printer:lpr://rzstud1.uni-karlsruhe.de:1234/queue1
(service:abstracttype:concretetype:URL)

scopes = rz, students, administrator
printer-name = bw600dpi
printer-model= HP LaserJet 20000
printer-location = Medienausgabe
color-supported = false

pages-per-minute = 30
sides-supported = one-sided, two-sided

Suche Um einen Dienst im Netzwerk zu finden, sendet ein UA einen *Service Request* an einen oder mehrere DAs. Dieser Service Request beinhaltet Attribut-Werte Paare, die den gewünschten Dienst genauer beschreiben. Sobald ein DA die Anfrage beantwortet hat kommuniziert der UA direkt mit dem SA und der DA wird nicht weiter benötigt.

Probleme Ein Nachteil von SLP ist die flache Architektur des Netzes, wodurch der Einsatz in einer ubiquitären Umgebung nur schwer realisiert werden kann. Dahingehende Verbesserungen sind in SLP implementiert, aber es ist deutlich erkennbar, dass der ursprüngliche Einsatz.b.ereich home- oder kleine business-Anwendungen sind. Eine relativ einfache Möglichkeit die Skalierbarkeit von SLP zu erhöhen ist es, die Anzahl von DAs im Netz zu erhöhen, wodurch sich die Last der Suchanfragen besser verteilt. Eine weitere Möglichkeit die Skalierbarkeit zu erhöhen ist es für DAs einen Wirkungsbereich, so genannte *Scopes* zu verwenden. Ein Scope ist ein einfacher String und wird innerhalb eines Service Templates angegeben (siehe Beispiel) wodurch dieser Service nur für Clients zur Verfügung steht, die auch dem gleichen Scope angehören. Die Wirkungsweise von Scopes ähnelt der Wirkung von Peergroups in JXTA-C, jedoch werden Scopes normalerweise nicht dynamisch erzeugt sobald sie benötigt werden. UAs können entweder Advertisements sammeln um eine Liste an Scopes zu erhalten, oder aber sie verwenden DHCP um Scopes zu finden auf die sie zugreifen dürfen.[5] Keiner dieser beiden Ansätze löst jedoch das Problem einer flachen Architektur. In einer Umgebung mit mehreren Millionen Geräten kann SLP daher, zumindest in der derzeitigen Netzstruktur, nicht eingesetzt werden. Eine Erweiterung um eine hierarchische Organisation wäre denkbar, ist aber meines Wissens zum jetzigen Zeitpunkt nicht implementiert.

2.4 JINI

Ein *Service* in Jini[3] kapselt eine gewisse Funktionalität die anderen Services, Programmen oder Benutzern zur Verfügung gestellt werden soll und von diesen genutzt werden kann. In dem Einführungsbeispiel des Paketdienstes wären dies unter anderem der Datenbankzugriff um Paketinformationen lesen bzw. schreiben zu können, aber auch die Funktionalität eines Druckers um Listen auszudrucken genauso wie ein möglicherweise installiertes GPS-Gerät im Fahrzeug zu dessen Ortsbestimmung. Benötigte Services werden zur Erfüllung einer speziellen Aufgabe je nach Bedarf in einen Zusammenschluß (eine sogenannte *Federation*) aufgenommen oder wieder entfernt. Um Kommunikation zwischen Services zu ermöglichen wird das *Service Protocol* eingesetzt, welches beliebig erweitert werden kann, um benötigte Anforderungen zu erfüllen. Grundlegende Services, die in Jini benötigt werden, sind:

1. Erzeugen eines Services
2. Finden eines Services (lookup)
3. Kommunikation zwischen Services (communication)
4. Verwendung von Services (use)

Netzstruktur Der *Lookup Service* ist notwendig, um andere Services innerhalb einer Federation überhaupt auffinden und anschließend nutzen zu können. Seine Arbeitsweise ähnelt dem Directory Agent in SLP. Um Anfragen nach einem gewissen Service beantworten zu können vergleicht der Lookup Service ein gewünschtes Interface mit den Interfaces von bei ihm registrierten Objekten. Findet er ein Objekt, welches das gewünschte Interface implementiert, so beantwortet er die Anfrage durch ein Service proxy Objekt. Dieses kann anschließend verwendet werden, um mit Hilfe von *RMI* (Remote Method Invokation) den bereitgestellten Service zu nutzen. Der Registrierungsvorgang eines neuen Services bei dem Lookup Service besteht aus zwei Schritten. In einem ersten Schritt muß ein Service-Erbringer (Service Provider) einen passenden Lookup Service finden, (Abbildung 5) bevor er sich bei diesem registriert (Abbildung 6) und von diesem Moment an von anderen Services gefunden werden kann. Hat ein Lookup Service eine Suchanfrage beantwortet (Abbildung 7), und somit zwischen einem Client und einem Service Provider vermittelt, wird er nicht weiter benötigt um den Dienst zu nutzen. Der Client verwendet von nun an das Service Object (Abbildung 8) welches er vom Lookup Service erhält, zur Nutzung des Service. Der Ablauf von Discovery eines Lookup Services bis hin zur Nutzung eines Services wird in den folgenden Abbildungen veranschaulicht.[3]

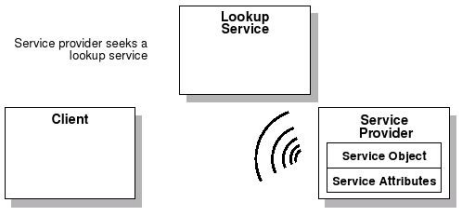


Abbildung 5. Suche eines Lookup service in JINI [3]

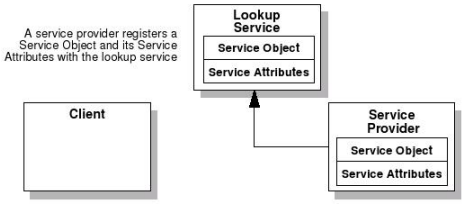


Abbildung 6. Registrierung eines Service-Objektes [3]

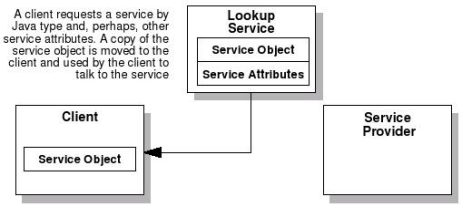


Abbildung 7. Suchanfrage durch einen Client [3]

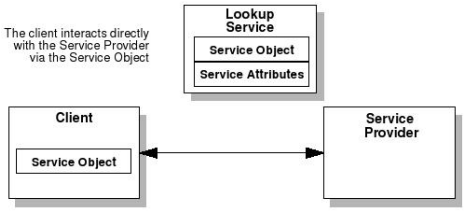


Abbildung 8. Suchanfrage durch einen Client [3]

Im Gegensatz zu SLP unterstützt Jini eine hierarchische Anordnung der Lookup Services und ermöglicht so den Einsatz in deutlich größeren Anwendungen. Jedoch stellt sich auch hier das in JXTA-C vorgestellte, nicht allgemein lösbare Problem der optimalen Einteilung. Je nach Wahl der Hierarchie können gewisse Anfragen effizienter bearbeitet werden oder erfordern ein Suche innerhalb des gesamten Netzes. Daher erfordert dieser Ansatz Hintergrundwissen über die Struktur und den Aufbau des Netzes um davon zu profitieren und um nicht versehentlich vermeidbare Engpässe einzubauen.

3 Vergleich der Ansätze

Die vorgestellten Architekturen unterscheiden sich hauptsächlich in der Netzwerkstruktur und somit in der Art und Weise wie sich Dienstgeber, Dienstnehmer und benötigte Infrastrukturdienste organisieren. Um auf weitere Unterschiede und deren Bewertung einzugehen sind in der Tabelle die wichtigsten Eigenschaften der Systeme gegenübergestellt.

3.1 Portierbarkeit

In einer ubiquitären Umgebung mit vielen zum Teil stark heterogenen Geräten liegt ein besonderes Augenmerk darauf ein sowohl programmiersprachen- als auch plattformunabhängiges System einzusetzen. Jini ist sehr stark an Java gekoppelt und setzt ein Java unterstützendes System voraus. Um einen Ausweg von dieser Abhängigkeit zu ermöglichen, unterstützt Jini auch einen Umweg über den Einsatz von Proxies. Dabei repräsentiert ein Gerät, das über eine Java Virtual Machine (JVM) verfügt, ein anderes ohne JVM und ermöglicht so die Nutzung von Jini, ohne zwangsläufig auf jedem Gerät Java zu fordern. Im Gegensatz zu Jini nutzt JXTA-C nicht die von einer speziellen Programmiersprache bereitgestellten Mechanismen bezüglich Sicherheit, Remote Method Invokation, Authorisierung oder andere. Die in JXTA enthaltenen Protokolle wurden unter anderem in Java und C implementiert und sind auf verschiedensten Geräten von Sensorknoten bis Großrechnern lauffähig.

3.2 Suchaufwand

Ein genereller Unterschied zwischen JXTA-C und den anderen Architekturen ist, dass JXTA-C bereits durch Wahl von Peergroups die Ressourcen in Klassen unterteilt. Anfragen werden anschließend nur von Rendezvous Peers innerhalb dieser Peergroup bearbeitet und auch nur zwischen diesen weitergeleitet.[2] Dies hat zur Folge, dass bei JXTA-C der Suchraum innerhalb einer Peergroup deutlich kleiner ist, die Antwort einer Anfrage jedoch auch auf einen Teil des Netzes beschränkt ist. Wenn ein Client sich bereits in der richtigenPeergroup befindet, ist dies sicherlich eine wünschenswerte Eigenschaft und keine Einschränkung. Jedoch bieten die anderen Architekturen den Vorteil, dass der Client sich nicht zuerst einer (im Allgemeinen unbekannten) Peergroup anschließen muß um eine

Tabelle 2. Übersicht der vorgestellten Systeme

	INS / Twine	**JXTA-C**	**SLP**	**Jini**
Skalierbarkeit	Advertisements auf mehrere Resolver verteilt	Unabhängige Peergroups	Begrenzt, nicht global einsetzbar	Hierarchische Anordnung der Lookup services
Architektur	Verteilt, flach	Verteilt, hierarchisch	Flach	Verteilt, hierarchisch
Hauptbestandteile	Resolver, Strand-Splitting	Peers, Relay/ Rendezvous Peers	UA / SA / DA	Lookup service, Client, Service
Dienstbeschreibung	XML; Strands	XML	Service URL, Service Template	Interfaces
Anfrageabarbeitung	Hashberechnung der Strands	Innerhalb Peergroup durch Rendezvous-Peers	Durch DA	Lookup Service, hierarchische Suche nach Interface
Discovery			DA Addresse durch: – Passiv Broadcast – Aktiv Request – DHCP	Discovery service um Lookup service initial zu finden
Konsistenz	Soft State; timeout im Netzkern länger	Soft State, Advertisements haben lifetime	Soft State	Soft State, Lease läuft ab
Suchraum	Gesamtes Netzwerk	Innerhalb Peergroups	Innerhalb Scope, falls verwendet	Konfigurierbar durch hierarchischen Lookup

Anfrage zu stellen. Dieser Vorteil wird jedoch erkauft durch einen im Schnitt höheren Suchaufwand, da eine Anfrage beispielsweise in INS/Twine über log(N) Resolver weitergeleitet werden muss [1], wohingegen in JXTA-C die Wahrscheinlichkeit hoch ist, eine häufige Anfrage bereits durch den ersten Rendezvous Peer beantworten zu können. Das Konzept der Rendezvous Peers stellt besonders für die Suche nach häufigen Advertisements einen effizienteren Mechanismus bereit als Hash-basierende verteilte indexierte Netzwerke.
Der Aufwand, um eine Anfrage beantworten zu können, hängt jedoch stark von der Art der Anfrage ab. So ist es in INS/Twine nur sehr aufwändig oder nicht möglich alle Ressourcen zu finden die sich innerhalb eines gewissen Umkreises befinden. Um dies zu ermöglichen wäre es notwenig Lokalität als Attribut in der Servicebeschreibung zu speichern. Sehr effizient kann jedoch die Frage nach allen Dienstgebern einer Klasse beantwortet werden, da sich alle zugehörigen Advertisements auf ein und demselben Resolver befinden. Beispielsweise in SLP

müsste ein DA rekursiv mit allen DAs innerhalb des Netzes kommunizieren um eine Liste aller verfügbaren SAs dieses Dienstes zu finden. Jini kann diese rekursive Suche durch Unterstützung von Hierarchien effizienter gestalten, vermeidet allerdings nicht mit allen Lookup Services zu kommunizieren.

3.3 Suchanfragen

Je nach Anwendung ist ein Vorteil von INS/Twine die Unterstützung von partiellen Suchanfragen, denn so ist es für einen Anwendungsentwickler nicht notwendig, diese gesondert zu implementieren. Im zweiten Einführungsbeispiel, bei dem ein Anwender nach einem lokal verfügbaren Drucker sucht, ist diese Eigenschaft häufig wünschenswert. Sucht der Anwender beispielsweise nach einem Drucker der über Duplex-Fähigkeit und eine gewisse Geschwindigkeit verfügt, wird dieser gefunden sofern er existiert. Existiert jedoch kein Gerät mit den gewünschten Eigenschaften, liefert der Resolver automatisch ein Gerät welches den Eigenschaften bestmöglich entspricht.

4 Ausblick

Die vorgestellten Architekturen, um innerhalb eines Netzes Dienste zu finden und Dienste bereitzustellen, unterscheiden sich in mehreren Punkten. Neben dem Aufbau des Netzes ist der Hauptunterschied die Art wie Dienstbeschreibungen auf verschiedenen Geräten gespeichert und anschließend wiedergefunden werden. Dabei ist deutlich sichtbar für welchen Zweck die Architekturen entwickelt wurden. So ist SLP, durch seine flache Architektur, eindeutig für eher kleinere lokale Einsatzzwecke entwickelt, wohingegen JXTA-C und INS/Twine für sehr große bzw. globale Netze entwickelt wurden. Abhängig von dem vorgesehenen Einsatzzweck kann daher auch nicht generell entschieden werden welcher Ansatz sich am besten eignet. Für ein kommendes Internet der Dinge mit vielen Millionen Geräten und globaler Ausdehnung des Netzes ist jedoch eine hierarchische Organisation des Netzes notwendig, um ein skalierendes System zu ermöglichen. Durch eine Einteilung in viele, weitestgehend voneinander unabhängige Peergroups reduziert JXTA-C den Suchraum erheblich. Allerdings stellt sich bei diesem Ansatz das Problem wie Peergroups möglichst optimal gebildet werden können. JXTA-C löst nicht das Problem eine passende Peergroup zu finden, sondern stellt Mechanismen bereit Peergroups zu verwalten und innerhalb einer Peergroup Dienste zu finden. INS/Twine umgeht die Einteilung des Netzes in viele kleinere Bereiche, jedoch ist dadurch der entstehende Suchaufwand höher. Eine generelle Aussage, welcher Ansatz sich auf Dauer durchsetzen wird, ist schwer zu treffen, jedoch ist eine Einteilung in kleinere Teilnetze flexibler und skaliert besser mit größeren Netzen. Unter der Annahme, dass das Problem der optimalen Gruppenerstellung und Gruppenfindung gelöst wird, stellt eine Einteilung in viele Peergroups vermutlich die flexibelste Architektur zur Verfügung.

Literatur

1. Balazinska, M., Balakrishnan, H., Karger, D.: Ins/twine: A scalable peer-to-peer architecture for intentional resource discovery (2002)
2. Traversat, B., Abdelaziz, M., Doolin, D., Duigou, M., Hugly, J., Pouyoul, E.: Project jxta-c: Enabling a web of things (2003)
3. Microsystems, S.: Jini technology architectural overview (1999)
4. Veizades, J., Guttman, E., Perkins, C., Kaplan, S.: Service location protocol (1997)
5. Guttman, E.: Service location protocol: Automatic discovery of IP network services. IEEE Internet Computing **3** (1999) 71–80

Middleware für Ubicomp

Melanie Denzel

Universität Karlsruhe (TH) ussz@stud.uni-karlsruhe.de

1 Einleitung

Ubicomp-Anwendungen zeichnen sich insbesondere dadurch aus, dass sie auf der Interaktion verschiedener Geräte basieren. Diese Geräte weisen eine hohe Heterogenität auf bezüglich zur Verfügung stehender Rechenkapazität, Speicherkapazität und Netzanbindung. Um einem Anwendungsprogrammierer trotzdem eine einheitliche Schnittstelle zur Kommunikation mit anderen Geräten bieten zu können, ist es erforderlich eine dafür angepasste Middleware einzusetzen. Im Rahmen dieser Seminararbeit werden bereits existierende Middlewarearchitekturen miteinander verglichen und hinsichtlich ihrer Eignung bewertet. Mit Devices Profile for Web Services (DPWS) wird zunächst ein Ansatz betrachtet, der auf standardisierten Web Services basiert. Die darauf folgende Architektur, Internet Communications Engine (ICE), baut auf CORBA auf und vereinfacht diese. Den Abschluss bildet die Middleware aus dem Projekt „Reconfigurable Ubiquitous Networked Embedded Systems“ (RUNES), ein auf Komponenten basierendes System.

2 Middleware allgemein

Verteilte Anwendungen werden auf mehreren Geräten ausgeführt, welche über ein Kommunikationsnetzwerk miteinander verbunden sind. Dabei können sowohl die Ausstattung der miteinander kommunizierenden Rechner als auch die verwendeten Netzwerktechnologien stark voneinander variieren. Die Entwicklung verteilter Anwendungen würde bei Berücksichtigung all dieser Unterschiede schnell sehr komplex werden. Daher wird zur Vereinfachung der Anwendungsprogrammierung eine zusätzliche Schicht zwischen dem Betriebssystem und der Anwendung eingesetzt: die Middleware. Sie überbrückt diese Heterogenität und verbirgt sie vor dem Programmierer. Zusätzlich kann die Middleware weitere Services anbieten, die die Anwendungsentwicklung vereinfachen.

2.1 Aufgaben von Middleware

Middleware verbirgt die Komplexität des verteilten Systems, so dass das System dem Anwendungsprogrammierer als zentriertes System erscheint. Daraus ergibt sich unter anderem die Aufgabe der Technologieunabhängigkeit. Damit Middleware auf Geräten mit unterschiedlicher Ausstattung (Hardware, Betriebssystem

oder Netzwerkanbindung) eingesetzt werden kann, muss sie die gleiche Funktionalität zur Verfügung stellen unabhängig von der darunterliegenden Technologie oder der verwendeten Programmiersprache. Die Middleware ermöglicht außerdem die operationale Interaktion zwischen verteilten Anwendungskomponenten, indem sie ein entsprechendes Modell zum entfernten Methodenaufruf bereitstellt. Weiterhin erlaubt die Middleware die Interaktion zwischen Komponenten, die sich in unterschiedlichen Adressräumen befinden. Verteilungstransparenz ist ebenfalls Aufgabe der Middleware. Diese ist notwendig, damit aus Anwendersicht kein Unterschied besteht zwischen einem lokalen und einem entfernten Prozeduraufruf. Die Middleware sorgt außerdem dafür, dass es dem Anwender verborgen bleibt, wenn einzelne Komponenten ausfallen oder kurzzeitig nicht zugreifbar sind, z.B. aufgrund einer schlechten Netzwerkverbindung.

2.2 Programmiermodelle

In Middlewaresystemen werden verschiedene Programmiermodelle unterstützt. Sie legen das Kommunikationsmodell und das Programmierparadigma der Middleware fest. Kommunikationsmodelle können synchron oder asynchron sein. Bei synchroner Kommunikation ist der Aufrufer während der Abarbeitung des Auftrags blockiert, während bei asynchroner Kommunikation der Aufrufer aktiv bleibt. Zu den verwendeten Programmierparadigmen gehören das prozedurale und das objektorientierte Paradigma. Zusätzlich kann das komponentenbasierte Paradigma als eine Erweiterung des objektorientierten Paradigmas betrachtet werden. In [5] werden darauf aufbauend drei Programmiermodelle definiert: *Entfernte Prozeduraufrufe* oder auch Remote Procedure Calls (RPC) verwenden synchrone Kommunikation und basieren auf dem prozeduralen Programmierparadigma. *Entfernte Methodenaufrufe* oder Remote Method Invocations (RMI) unterscheiden sich von RPCs dadurch, dass sie statt auf dem prozeduralen auf dem objektorientierten Paradigma aufbauen. Auf asynchrone Kommunikation baut hingegen das *nachrichtenorientierten Modell* auf. Dieses Modell verwendet keine Methoden- oder Prozeduraufrufe, sondern verpackt diese in Nachrichten und ist somit unabhängig von Programmierparadigmen. Das Publish/Subscribe Modell kann als Sonderfall des nachrichtenorientierten Modells gesehen werden.

3 Middleware für Ubicomp

Für ubiquitäre Anwendungen steht die Interaktion der Geräte untereinander stark im Vordergrund. Das bedeutet, dass es sich bei Anwendungen aus dem Bereich „Ubiquitous Computing“ immer auch um verteilte Anwendungen handelt. Deshalb ist auch bei ubiquitären Anwendungen der Einsatz einer Middleware von Vorteil. Im Vergleich zu allgemeinen verteilten Anwendungen weisen ubiquitäre Anwendungen bestimmte Eigenschaften auf, die es erlauben, spezielle Anforderungen an die Middleware abzuleiten.

3.1 Eigenschaften von Ubicomp-Anwendungen

Ein ubiquitäres System zeichnet sich aus durch eine hohe Heterogenität der beteiligten Geräte. Die Geräte reichen von Laptops mit großer Speicherkapazität und Rechenleistung über PDAs bis hin zu einfachen Sensorknoten, die nur mit einem minimalen Betriebssystem ausgestattet sind. Zu dieser Heterogenität der Geräte kommt noch die meist große Zahl an teilnehmenden Geräten hinzu. Auch die in ubiquitären Systemen verwendeten Netze sind sehr unterschiedlich. So treten unter anderem viele drahtlose Kommunikationsmechanismen wie Bluetooth, IrDa oder WLAN auf. Eine weitere typische Eigenschaft ist die Spontaneität der Zusammenschließung der Geräte. Oft handelt es sich um ad-hoc Netze, die nur kurzzeitig existieren und nicht unbedingt eine Verbindung zu einer festen Infrastruktur (Backbone) haben. Dies ist darauf zurückzuführen, dass die Geräte selbst meist mobil sind und somit nicht stationär an ein Netz gebunden sind, sondern in neue Netze eintreten können und diese auch wieder verlassen können. Von dieser physischen Mobilität zu unterscheiden ist die logische Mobilität. Hierbei wechselt nicht ein Gerät seinen Ort, sondern ein Service wandert von einem Gerät zu einem anderen. Dies kann zum Beispiel dann der Fall sein, wenn ein Gerät ausfällt. Wenn möglich übernimmt dann ein anderes Gerät dessen Service. Dieser Vorgang wird mit dem Begriff Service-Migration bezeichnet.

3.2 Anforderungen an Ubicomp-Middleware

Aus den im letzten Abschnitt aufgezhlten Eigenschaften, lassen sich mehrere Anforderungen an eine Middleware für Ubicomp-Anwendungen ableiten. Diese sollen in den nächsten Abschnitten erlutert werden.

Unterstützung heterogener Geräte Die Middleware muss eine große Bandbreite an Geräten unterstützen. Ein weit verbreiteter Ansatz ist der Einsatz von Proxys oder Gateways, die der Anwendung vorgeschaltet sind und somit Aufrufe für ein bestimmtes Gerät anpassen können. Außerdem bieten Gateways die Möglichkeit Services von mehreren Geräten zu bündeln, so dass der Zugriff auf mehrere Services über dieses Gateway gesteuert wird.

Unterstützung heterogener Netze Um auf verschiedenen Netzen einsetzbar zu sein, muss die Middleware unterschiedliche Transportmechanismen unterstützen. Falls eine Anwendung über Netzwerkgrenzen hinweg ausgeführt werden soll, also zum Beispiel auch über das Internet, muss auch die Problematik von Firewalls beachtet werden. Eine Firewall schützt die dahinter liegenden Netzwerke vor ungewolltem Zugriff, indem sie nur bestimmte Protokolle, wie beispielsweise HTTP, erlaubt und alle anderen Kommunikationsversuche blockiert. Eine Möglichkeit der Umgehung von Firewalls stellt das Tunneling dar. Dabei werden Middleware-Nachrichten zum Beispiel als HTML-Dokumente verpackt und über HTTP versendet.

Hohe Skalierbarkeit Aufgrund der großen Anzahl an Geräten und der Mobilität dieser Geräte, kann nie genau vorhergesagt werden wie viele Geräte zu einer bestimmten Zeit an einem ubiquitären System teilnehmen und dessen Services nutzen. Daher muss die Middleware eine hohe Skalierbarkeit aufweisen.

Einfachheit Da ubiquitäre Anwendungen auch auf Sensorknoten ausgeführt werden, muss die Middleware relativ einfach gehalten sein, um auch dort implementierbar zu sein.

Im Folgenden werden die drei Middlewareansätze DPWS, ICE und RUNES genauer beschrieben und bewertet. Die Bewertung erfolgt anhand der im letzten Kapitel aufgestellten Anforderungen an Middlewaresysteme für Ubicomp-Anwendungen. Außerdem wird auf die Fragestellung eingegangen, ob die betrachteten Middlerwarelösungen generisch einsetzbar sind für Client/Server- und Peer-to-Peer-Architekturen und ob eine Kommunikation in ad-hoc Netzen unterstützt wird.

4 Devices Profile for Web Services (DPWS)

DPWS fasst mehrere Web Services zusammen, die damit die Funktionalität einer Middleware zur Verfügung stellen. Daher folgt zunächst eine knappe Erklärung von Web Services und erst im Anschluss wird genauer auf den eigentlichen Ansatz von DPWS eingegangen.

4.1 Web Services

[5] definiert Web Services als Softwaresysteme, die die Maschine-zu-Maschine Kommunikation über das Internet ermöglichen. Der Zugriff auf Web Services wird über eine Schnittstellenbeschreibung festgelegt und erfolgt über SOAP-Nachrichten. Da SOAP aber eigentlich kein Protokoll ist, sondern nur das Nachrichtenformat definiert, werden SOAP Nachrichten über ein Kommunikationsprotokoll wie zum Beispiel HTTP, FTP oder SMTP ausgetauscht.

4.2 Aufbau von DPWS

DPWS ist eine Spezifikation zur service-basierten Kommunikation zwischen Geräten mit eingebetteten Systemen. Dabei nutzt DPWS ausgewählte Web Services um damit folgende Dienste anzubieten:

- Sicheres Senden von Nachrichten zu und von einem Web Service
- Dynamisches Suchen und Finden von Geräten
- Austausch von Metadaten
- Publish-Subscribe Eventing Service

[6] enthält eine Übersicht über die Web Services Standards, die für die DPWS Spezifikation verwendet werden. Dazu gehört die Beschreibungssprache *WSDL (Web Service Description Language)*, die der Definition der Schnittstellen von Web Services dient. *SOAP* bietet einen Standard für das Nachrichtenformat zum Zugriff auf die Sevices, das zu den jeweiligen WSDL-Definitionen konform ist. Die zur Konstruktion dieser Nachrichten benötigten Datentypen werden mit Hilfe von *XML Schema* definiert. Die Schnittstellendefinition kann zusätzlich um Policies z.B. zur Zugriffsbeschränkung erweitert werden, welche durch *WS-Policy* angegeben werden können. *WS-Addressing* legt fest, dass alle Adressierungsinformation der Nachricht im SOAP Header untergebracht wird. Dadurch wird der Inhalt der Nachricht von der Transportinformation entkoppelt und erlaubt komplexere Austauschkonzepte als das einfache Request-Response-Modell von HTTP. Um Metadaten von Services (Beschreibung, Schema und Policy) zu erhalten kann *WS-MetadataExchange* verwendet werden. Optional kann *WS-Security* hinzugenommen werden um Sicherheitsmechanismen einzubauen.
Zu diesen Kernstandards fügt DPWS noch Protokolle für Discovery und Events hinzu. *WS-Discovery* erlaubt die Suche und das Advertising von Services bzw. Geräten in ad-hoc Netzen. Die Suche von Services geschieht über das Senden von Multicast-Nachrichten, sog. Probes, in denen der gesuchte Service spezifiziert wird. Entspricht ein Gerät dem gesuchten Typ, sendet es direkt eine Antwort an den Anfrager zurück. Ein Gerät, das Services anbietet kann diese auch bekannt machen, indem es bei Betreten eines neuen Netzes eine Hello Nachricht an die Multicast-Adresse schickt. Verlässt das Gerät das Netz, meldet es sich mit einer Bye-Nachricht wieder ab. *WS-Eventing* definiert ein Publish-Subscribe Protokoll. Dadurch wird es einem Web Service (Subscriber) ermöglicht, sich bei einem anderen Web Service (Publisher) für ein bestimmtes Event zu registrieren. Bei Eintreten dieses Events schickt der Publisher eine Nachricht (Notification) an den Subscriber.
Der vollständige Protokollstack sieht damit folgendermaßen aus:

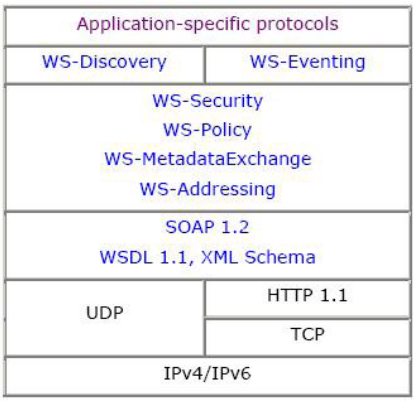


Abbildung 1. Protokollstack von DPWS

In [6] wird außerdem eine Möglichkeit beschrieben, wie man „primitive“ Geräte, die nicht DPWS-fähig sind, integrieren kann. Dazu werden Device Gateways verwendet, die sämtliche DPWS-Funktionalität für diese Geräte übernehmen. Sie veröffentlichen zum Beispiel deren Schnittstellen und leiten Nachrichten an diese Geräte weiter. Von außen merkt also ein Teilnehmer nicht, dass er sich über ein Gateway mit dem Gerät unterhält.

4.3 Einordnung

Eine Implementierung der DPWS-Protokolle mit Ausnahme von WS-Security wurde in [6] für die folgenden Betriebssysteme bereitgestellt: Linux, Windows, Windows CE, ThreadX und Quadros. Unterstützt wurde dabei die Programmiersprache C. Grundprinzip der Interaktion ist ein service-orientiertes Modell. Alles wird als Service angesehen, auf den über eine Schnittstelle zugegriffen werden kann. Der Nachrichtenaustausch kann sowohl synchron als auch asynchron, z.B. durch Events oder auch nur in eine Richtung als one-way-Nachricht erfolgen.

4.4 Ist DPWS generisch einsetzbar?

DPWS unterteilt Geräte in „controlling devices“ und „controlled devices“, also Geräte, die gesteuert werden und solche, die andere Geräte steuern. Dies entspricht im Prinzip einer Client/Server-Architektur. Allerdings kann ein Gerät sowohl die Rolle eines controlling als auch eines controlled device einnehmen, womit auch peer-to-peer-Interaktion möglich ist.
Durch die Unterstützung des Discovery von plug-and-play-Geräten ist DPWS auf jeden Fall auch für die Ausführung in ad-hoc Netzen geeignet und nicht auf Infrastruktur-basierte Netze beschränkt.

4.5 Eignung für Ubicomp

Unterstützung heterogener Netze Da DPWS auf IPv4 bzw. IPv6 aufbaut, kann DPWS nur in Verbindung mit IP-basierten Netzen verwendet werden. Das bedeutet, dass keine Infrarot- oder Bluetooth-Netze unterstützt werden. Allerdings kann DPWS auf einer großen Bandbreite an lokalen Netzen wie LAN und WLAN ausgeführt werden. Aber auch eine Ausführung über lokale Netze hinweg wird unterstützt. Hierzu bietet DPWS einen WS-Discovery-Modus bei dem ein Discovery Proxy eingesetzt wird. Dieser verbindet das lokale Netz mit anderen Netzen und unterdrückt das Senden von Multicast Nachrichten. Dadurch erreicht die Anwendung eine höhere Skalierbarkeit. Das Firewall-Problem besteht bei der Verwendung von DPWS nicht, da als Kommunikationsprotokoll HTTP verwendet wird, welches von Firewalls nicht blockiert wird.

Unterstützung heterogener Geräte DPWS ist auf einer Vielzahl von Geräten einsetzbar und ist nicht auf ressourcenreiche Geräte beschränkt. Insbesondere können auch nicht DPWS-fähige Geräte über Gateways integriert werden.

Skalierbarkeit Zur Skalierbarkeit bei großen Netzen trägt wesentlich die Einführung eines Discovery Proxies bei. Durch das Unterdrücken der Multicast-Nachrichten wird der Datenverkehr deutlich reduziert und eine Ausdehnung des Discoveryprozesses auf andere Netze ermöglicht. Bei Ausfall eines Discovery Proxies fallen die von ihm verwalteten Geräte sofort wieder in den Standard-Discovery-Modus zurück und bleiben so innerhalb ihres lokalen Netzes verfügbar.

Einfachheit Nach den ersten Tests, die von den Autoren von [6] durchgeführt wurden, beschränken sich der statische Speicherbedarf auf 500 KB und der dynamische auf 100KB bei einem Prozessor mit 44 MHz ARM7 TDMI. Dies beinhaltet sowohl den Bedarf für das Betriebssystem (ThreadX) und den TCP/IP Protokollstack als auch die DPWS-Software. Damit ist DPWS auch auf Geräten mit beschränkten Ressourcen einsetzbar.

5 Internet Communications Engine (ICE)

ICE baut auf der verbreiteten Middleware-Architektur CORBA auf, weshalb im Folgenden zuerst der Aufbau von CORBA genauer erläutert wird. Anschließend werden dann der Aufbau und die Eigenschaften von ICE betrachtet.

5.1 Common Object Request Broker Architecture (CORBA)

CORBA spezifiziert eine Middleware für objektorientierte verteilte Anwendungen. Eine zentrale Rolle spielt hierbei der Object Request Broker, welcher Aufrufe von Client- an Serverobjekte übermittelt. Die nachfolgende Abbildung veranschaulicht dieses Vorgehen.

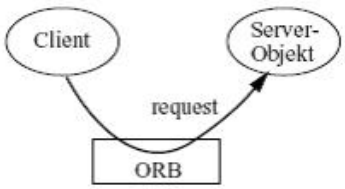


Abbildung 2. Aufrufbermittlung über den ORB

Um die genauere Arbeitsweise von CORBA zu erläutern, werden im Folgenden die einzelnen Komponenten und deren Aufgaben erklärt.

Der *ORB* ist die Kernkomponente, die zwischen den einzelnen Objekten Aufrufe vermittelt. Hierfür ist es wichtig, dass Objekte eine eindeutige Adresse haben. Die *Interface Definition Language (IDL)* dient der Beschreibung der Signaturen der Methoden eines Objekts und definiert benötigte Datentypen.

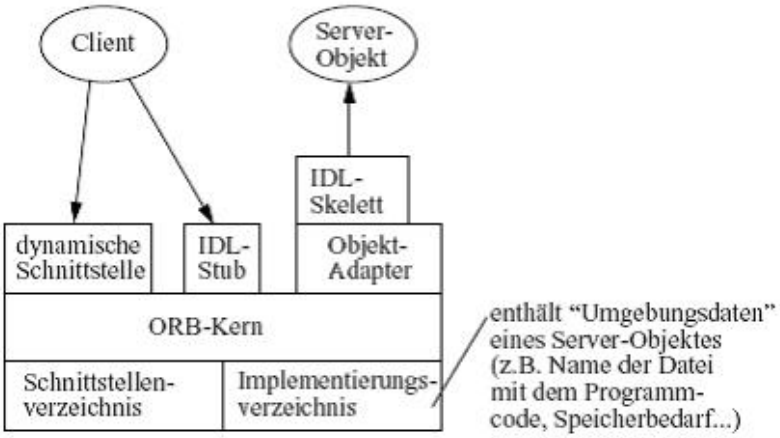


Abbildung 3. Komponenten der CORBA-Architektur

Bei statischen Aufrufen muss die Schnittstellenbeschreibung vor Ausführung bekannt sein. Der IDL-Compiler generiert dann aus dieser Schnittstellenbeschreibung die programmspezifische Implementierung der Objektschnittstelle. Über den dadurch entstandenen clientseitigen Code (*Stub*) kann der Client auf die entsprechenden entfernten Methoden zugreifen. Der auf der Serverseite generierte Code wird mit *Skeleton* bezeichnet. Hauptaufgabe der Stubs und Skeletons ist das Anpassen der Formate (Marshalling, Unmarshalling) und die Aufrufvermittlung. Um auch dynamische Aufrufe zu unterstützen, wird das *Schnitstellenverzeichnis (interface repository)* verwendet. Es speichert IDL-Schnittstellen und andere Zusatzinformationen zum Zugriff auf Serverobjekte. Über die *dynamische Schnittstelle (Dynamic Invocation Interface)* des ORB hat der Client Zugriff auf das Schnittstellenverzeichnis. Dort sucht er die gewünschte Schnittstellenbeschreibung und kann dann den Aufruf entsprechend dieser Beschreibung generieren und an den ORB weiterleiten. Der *Objektadapter* verwaltet den Lebenszyklus des Serverobjekts, also das Initialisieren, Passivieren, Aktivieren und das Löschen. Er ist für die Generierung eindeutiger Objektreferenzen zuständig und leitet zur Laufzeit Aufrufe an das Serverobjekt weiter. Das *Implementierungsverzeichnis (Implementation Repository)* unterstützt den Objektadapter im Management der Serverobjekte, indem es zu jedem verwalteten Objekt Informationen speichert.
CORBA definiert außerdem eine Menge an verfügbaren Services, die unabhängig von der Anwendung sind. Dazu gehören unter anderem ein Naming Service, ein Event- und Notification- Service, ein Transaction-Service und ein Security-Service.

5.2 Aufbau von ICE

ICE basiert auf der Middleware-Plattform CORBA, bietet allerdings einige Vereinfachungen und Verbesserungen des Objekt-Modells von CORBA. [4] enthlt die genaue Beschreibung der Komponenten der ICE-Middleware und eine Auflistung der Vorteile dieser Spezifikation im Vergleich zu CORBA. Im Folgenden soll kurz auf einige dieser Verbesserungen eingegangen werden.

Vereinfachung des Typsystems Nach Ansicht von M. Henning [3] beinhaltet das Typsystem von CORBAs IDL eine zu große Bandbreite an Datentypen. Diese sind zum Teil unnötig oder redundant oder können auch zu Problemen führen, da sie nicht in allen Programmiersprachen oder von allen CPUs unterstützt werden. So bietet beispielsweise IDL den Datentyp unsigned integer, welcher aber in JAVA nicht existiert. SLICE, die Spezifikationssprache von ICE, hingegen lässt diese unnötigen Typen weg und vereinfacht somit das Typsystem.

Reduzierung der Laufzeit-API Eine weitere Verbesserung betrifft die Komplexität der Laufzeit-API. In CORBA ist die API oft sehr komplex, was deren Benutzung stark erschwert. In ICE wurde die API deutlich verkürzt. So wurde beispielsweise die API des CORBA-Objektadapters von 211 Zeilen IDL-Spezifikation auf 29 Zeilen Slice-Definitionen in ICE reduziert.

Erweiterung der Aufrufsemantik CORBA unterstützt synchrone, asynchrone und one-way Aufrufsemantiken. ICE erweitert diese um Datagram Invocation und Batched Invocation. Datagram Invocation unterstützt one-way Aufrufe über UDP Datagramme. Dies ist z.B. nützlich zur Übertragung von Events, da UDP als verbindungsloser Dienst bessere Leistung einbringt, die Skalierbarkeit erhöht und weniger Ressourcen beansprucht. In ICE ist es durch batched invocation außerdem möglich one-way und datagram invocations auf Clientseite zu speichern bis die Anwendung den Flush-Befehl gibt. Dadurch genügt eine Nachricht für das Senden mehrerer Aufrufe, während in CORBA für jeden Aufruf eine Request- und eine Replynachricht über das Netz geschickt werden.

Möglichkeit zur Kompression ICE bietet eine „on-the wire“-Kompression, welche sinnvoll ist bei Verbindungen mit geringer Bandbreite.

5.3 Einordnung

ICE basiert wie CORBA auf dem objektorientierten Programmierparadigma. Es bietet Möglichkeiten zur synchronen, asynchronen (über Call-Back Objekte) und one-way Kommunikation. Es unterstützt derzeit lediglich die Betriebssysteme Windows und Linux. Allerdings wird eine Menge an Programmiersprachen unterstützt. Dazu gehören C++, C#, Java, Visual Basic, .Net, Python und PHP.

5.4 Ist ICE generisch einsetzbar?

ICE ist vorrangig als Middleware für Client/Server-Anwendungen gedacht. Es werden zwar die beiden Fälle betrachtet, dass ein Server als Client agiert, wenn er eine Clientanfrage an einen anderen Server weiterleitet bzw. dass ein Client als Server agiert, wenn er dem Server ein Callback-Objekt zur Verfügung stellt, aber auf eine explizite Peer-to-Peer Kommunikation wird nicht eingegangen. Prinzipiell sollte es allerdings möglich sein, ICE auch für Peer-to-Peer Interaktionen zu verwenden, indem jeder Teilnehmer sowohl die serverspezifischen Elemente wie den Objektadapter als auch die clientspezifischen Elemente wie die dynamische Schnittstelle zum ORB implementiert.
ICE kann auch für ad-hoc Netze eingesetzt werden, falls es den Trading-Service von CORBA unterstützt. Dieser ermöglicht das Advertisement und Discovery von Services. Allerdings müssen dazu auch alle Teilnehmer diesen Service implementiert haben, was von CORBA nicht vorausgesetzt wird.

5.5 Ist ICE für Ubicomp-Anwendungen geeignet?

Unterstützung heterogener Netze ICE ist anwendbar für UDP- und TCP-basierte Netze. Außerdem kann es über SSL als Transportschicht eingesetzt werden. Das bedeutet aber, dass ICE nicht für Bluetooth oder IrDA verwendet werden kann. Allerdings unterstützt ICE die Kompression der übermittelten Nachrichten und verbessert damit die Kommunikation über Verbindungen mit einer niedrigen Bandbreite. Um über Firewalls zu kommunizieren verwendet ICE die vom Client zum Server aufgestellte Verbindung bidirektional. Schickt also ein Client einen Aufruf mit Call-Back an einen Server, dann kann der Server nach Beenden seiner Ausführung über dieselbe Verbindung einen Aufruf an das Call-Back-Objekt schicken.

Unterstützung heterogener Geräte ICE ist nicht für die Ausführung auf Geräten mit beschränkten Ressourcen gedacht. Dies zeigt sich unter anderem daran, dass nur Geräte mit den Betriebssystemen Windows und Linux unterstützt werden. Es fehlen Betriebssysteme wie Windows CE oder Contiki OS, die für Geräte mit geringer Kapazität verwendet werden. Trotzdem bietet ICE im Vergleich zu CORBA einige Verbesserungen hinsichtlich der Komplexität und damit sicherlich auch hinsichtlich des Speicherbedarfs. Für die Ausführung auf Sensorknoten erscheint ICE allerdings zu komplex.

Skalierbarkeit Um mit einer größeren Zahl an Clients zurechtzukommen bietet ICE die Möglichkeit, mehrere Server zu einer Federation zusammenzuschließen. Dabei bieten alle Server den gleichen Dienst an und die Last wird gleichmäßig auf alle Rechner verteilt. Zusätzlich unterstützt ICE die Verwendung von UDP als Transportprotokoll. Dadurch verringert sich der entstehende Ressourcenbedarf erheblich im Vergleich zum Einsatz von TCP-Verbindungen. Die Auswirkungen davon sind ganz besonders deutlich, wenn ICE Events an eine sehr große Zahl an

Clients übermitteln muss. Der gesamte Vorgang des Verbindungsaufbaus bzw. Verbindungsabbaus entfällt hierdurch.

Einfachheit Leider erscheint ICE noch relativ komplex, so dass es nicht direkt auf Sensorknoten einsetzbar ist. Außerdem wird keine Möglichkeit genannt, wie ressourcenarme Geräte integriert werden können. Allerdings haben die sich die Autoren von [3] das Ziel gesetzt, auch eine Version von ICE zu entwickeln, die für eingebettete Umgebungen verwendet werden kann.

6 Reconfigurable Ubiquitous Networked Embedded Systems (RUNES)

RUNES ist ein europäisches Projekt, welches zum Ziel hat, die Entwicklung von verteilten, heterogenen, vernetzten eingebetteten Systemen zu ermöglichen. Innerhalb dieses Projekts, das momentan noch nicht ganz abgeschlossen ist, wurde auch eine Middleware definiert, die die Interaktion der einzelnen Systeme erleichtert.

6.1 Aufbau

Die RUNES Middleware ist komponenten-basiert. Dies bedeutet, dass Implementierung und Schnittstelle voneinander entkoppelt sind und somit die Implementierung einer Komponente angepasst werden kann ohne deren Schnittstelle zu verändern. Dies ist insbesondere für die dynamische Anpassung der Middlewarekomponenten wichtig. [1] gibt eine Übersicht über die Elemente der RUNES Middleware, während [2] ausführlicher auf der Aufbau der RUNES Middleware eingeht. Die einzelnen Elemente sind in nachfolgendem Bild veranschaulicht.

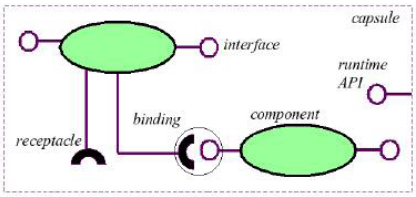


Abbildung 4. Elemente des RUNES Komponentenmodells

Die *Capsule* ist in jedem RUNES System genau einmal enthalten. Sie repräsentiert einen Adressraum, in dem alle Komponenten dieses Systems ablaufen. Sie bietet eine Laufzeit-API zum dynamischen Laden, Unloading, Instantiieren und Löschen von Komponenten. Auf verschiedenen Geräten kann die Capsule unterschiedlich implementiert werden. Auf einem PC oder PDA könnte sie zum Beispiel als Prozess implementiert werden und auf einem Sensorknoten

als RAM Chip.
Eine *Komponente* kapselt eine bestimmte Funktionalität wie zum Beispiel einen Transportmechanismus wie TCP oder UDP. Ihr können Attribute als Schlüssel-Wert-Paare zugeordnet werden. Über Interfaces und Receptacles interagiert die Komponente mit anderen Komponenten. *Interfaces* definieren die Signatur und die Datentypen der Methoden, die von der Komponente zur Verfügung gestellt werden. Sie werden mit Hilfe der Schnittstellenbeschreibungssprache OMG IDL angegeben und sind damit unabhängig von der Programmiersprache. Das *Receptacle* beschreibt die verwendeten Interfaces von anderen Komponenten und gibt damit explizit vorhandene Abhängigkeiten an. Durch ein Binding werden Receptacle und Interface zweier Komponenten verbunden. Dieses Binding kann auch verteilt erfolgen, so dass Receptacle und Interface von zwei entfernten Komponenten verbunden werden können.
Ein *Komponenten-Framework (CF)* besteht aus einer Gruppe von Komponenten, die gemeinsam eine Aufgabe erfüllen. Dabei ist es auch möglich zur Laufzeit leichtgewichtige Komponenten, sogenannte Plug-Ins, hinzuzufügen um die Funktionalität des CFs zu modifizieren oder zu erweitern.
RUNES definiert einige CFs, die jeweils bestimmte Services zur Verfügung stellen. *Local OS Services* beinhalten zum Beispiel typische Funktionalität, die eigentlich vom Betriebssystem bereitgestellt wird. Die *Network Services* erlauben über Plug-Ins verschiedene Kommunikationsmöglichkeiten und bieten eine generische API für den Zugriff (Bsp.: send(message, address) für unicast, sendToAll(message) für broadcast). Dabei wird sowohl infrastruktur-basierte als auch ad-hoc Kommunikation unterstützt. Durch *Interaction Services* wird eine erweiterbare Menge an Interaktionsparadigmen unterstützt. Dazu gehören zum Beispiel Publish/Subscribe mit Event Notification, RPC und Tuple Spaces. Über *Advertising* und *Discovery Services* können verschiedene Strategien wie zum Beispiel UPnP oder SLP zum dynamischen Suchen von Services eingesetzt werden. Weiterhin existieren *Location Services* zur Bestimmung der physischen bzw. der logischen Position und *Coordination Services* zur Koordination der Sensoren.

6.2 Einordnung

Die RUNES Middleware ist komponenten-basiert. Sie bietet sowohl synchrone als auch asynchrone Kommunikation. Durch Interaction Services können Interaktionsparadigmen beliebig ausgetauscht und erweitert werden, wodurch eine breite Masse an Paradigmen unterstützt wird. Wie unter [Homepage www.ist-runes.org] angegeben werden derzeit die Plattformen Unix/C, Contiki OS, Java Virtual Machine und Erlang Virtual Machine unterstützt. Als Programmiersprachen werden bisher nur Java und C angeboten.

6.3 Ist RUNES generisch einsetzbar?

Durch die Möglichkeit mit Plug-Ins die Funktionalität der Middleware anzupassen, ist die Middleware sehr generisch einsetzbar. Dabei werden sowohl Client/Server-Anwendungen unterstützt als auch Peer-to-Peer-Anwendungen. Genauso ist es

möglich die Middleware für ad-hoc Netze einzusetzen, da der Networking Service ebenfalls um beliebige Komponenten erweiterbar ist.

6.4 Eignung für Ubicomp

Unterstützung heterogener Netze RUNES Middleware kann sehr heterogene Netze als Transportmedium verwenden. Dies ist wieder auf die Auswechselbarkeit der Netzwerk-Komponenten zurückzuführen. Auf das Firewallproblem wurde in [1,2] leider nicht eingegangen.

Unterstützung heterogener Geräte Die RUNES Middleware ist für eine große Bandbreite an Geräten konzipiert. Sie unterstützt sowohl Sensorknoten (Contiki OS) als auch PDAs und Laptops (derzeit über Java).

Skalierbarkeit Die Skalierbarkeit dieser Middleware hängt stark von der tatsächlichen Implementierung der einzelnen Komponenten und der Verfügbarkeit verschiedener Plug-Ins ab. So gibt die Middleware in den meisten Fällen nicht die genaue Strategie vor, die für einen Service verwendet werden soll, sondern legt lediglich das Interface des Services fest. So kann zum Beispiel der Discovery Service auf Basis von SLP oder UPnP implementiert werden oder sogar beide Strategien zur Verfügung stellen und abhängig von der darunter liegenden Netztopologie entscheiden, welche davon vorzuziehen ist. Dadurch wird die Middleware sehr flexibel und kann auf Veränderungen in der Umwelt reagieren. Genauso kann über die Konfiguration der Network Service Komponenten beeinflusst werden, wie zum Beispiel eine Broadcast-Nachricht verbreitet wird. Das Interface definiert lediglich, dass über die Methode sendToAll(message) die Nachricht als Broadcast verschickt wird. Ob aber die Empfänger der Nachricht diese einfach an sämtliche Nachbarn weiterleiten oder ob die Verbreitung der Nachricht über Routingprotokolle gesteuert wird, ist abhängig von der gewählten Implementierung.

Einfachheit Durch das Komponentenmodell ist es möglich die Middleware relativ einfach zu halten, indem auf einem Sensorknoten nur die Teile der Middleware verwendet werden, die auch tatsächlich benötigt werden. Durch dynamisches Laden neuer Komponenten ist es gegebenenfalls möglich die Funktionalität der Middleware zur Laufzeit anzupassen.

7 Vergleich

Die Tabelle 1 bietet noch mal einen Überblick über die bereits herausgearbeiteten Eigenschaften der verschiedenen Middlewareansätze.

Beim Vergleich fällt auf, dass sich die verschiedenen Middlewarelösungen hinsichtlich der zur Verfügung gestellten Menge an Services kaum unterscheiden. So bieten zum Beispiel alle einen Discovery Service an oder die Möglichkeit zur

Tabelle 1. Übersicht über Eigenschaften der Middlewareansätze

	DPWS	**ICE**	**RUNES**
unterstützte Betriessysteme	Linux, Windows, Windows CE, ThreadX, Quadros	Unix, Windows	Java-Virtual Machine, Unix/C, Contiki OS, Erlang Virtual Machine
unterstützte Programmiersprachen	C	C++, C#, Java, PHP, Python, Visual Basic, .Net	Java, C
Programmierparadigma	service-orientiert	objekt-orientiert	komponenten-orientiert
Kommunikation	asynchron, synchron	asynchron, synchron	beliebig

Event Notification. Auch unterstützen alle sowohl synchrone als auch asynchrone Kommunikation.
Unterschiede zeigen sich vor allem bei der Unterstützung verschiedener Plattformen. Sowohl RUNES als auch DPWS eignen sich sehr gut für den Einsatz auf heterogenen Geräten. Sie unterstützen eine Vielzahl an Betriebssystemen und können auch auf Geräten mit beschränkten Ressourcen eingesetzt werden. Bei RUNES wird dies insbesondere durch die Komponenten-Architektur unterstützt, die es offen lässt, welche Services angeboten werden und wie einzelne Services implementiert werden. Dadurch kann die Middleware stark an unterschiedliche Gerätetypen und deren Anforderungen angepasst werden. DPWS dagegen ermöglicht durch Verwenden von Device Gateways die Integration von primitiven Geräten, die DPWS nicht unterstützen. ICE ist diesbezüglich wenig variabel was den Umfang der Middleware betrifft, soll aber noch für den Einsatz auf eingebetteten Systemen angepasst werden.
Die Unterstützung für heterogene Netze ist nur bei RUNES tatsächlich gegeben. Sowohl ICE als auch DPWS bauen auf TCP oder UDP auf und bieten damit keine Unterstützung für Bluetooth- und Infrarotverbindungen. Aber selbst bei Netzen, die UDP und TCP unterstützen, wie zum Beispiel WLAN, kann es zu Schwierigkeiten kommen aufgrund der beschränkten Bandbreite. Prinzipiell kann DPWS über Verbindungen mit niedriger Bandbreite eingesetzt werden, aber durch die Verwendung von XML werden die zu übermittelnden Nachrichten deutlich größer als dies bei Protokollen der Fall ist, die kein XML verwenden. ICE hingegen vereinfacht die Benutzung von Verbindungen mit geringer Bandbreite durch die optionale Kompression der Nachrichten.
Insgesamt lässt sich sagen, dass RUNES wohl den flexibelsten Ansatz darstellt was die Unterstützung der Heterogenität betrifft. Allerdings ist das RUNES Projekt noch nicht abgeschlossen, weshalb noch keine komplette Implementierung vorhanden ist und eine gesamte Bewertung daher noch schwierig ist. ICE hingegen basiert auf der weit verbreiteten Middleware CORBA und kann daher auf Erfahrungen im Einsatz dieser Middleware zurückgreifen. Die Entwickler von ICE haben bewährte Konzepte übernommen und insbesondere Probleme von

CORBA analysiert und diese in ihrer Spezifikation behoben. Daher ist anzunehmen, dass ICE in der Praxis deutlich zuverlässiger funktioniert als dies bei der noch neuen RUNES Middleware der Fall sein dürfte. DPWS benutzt bereits existierende Standards für seine Middleware und hat dadurch sichergestellt, dass Implementierungen verschiedener Hersteller miteinander kooperieren können.

8 Zusammenfassung und Ausblick

Ubicomp-Anwendungen stellen besondere Anforderungen an eine Middleware. Allerdings unterscheiden sich auch Ubicomp-Anwendungen stark untereinander, sodass es schwierig ist, eine Middleware zu finden, die alle Einsatzbereiche abdeckt. RUNES scheint hierfür geeignet zu sein, da es sehr flexibel ist und sich an unterschiedliche Bedingungen anpassen lässt. Für eine gegebene Anwendung können benötigte Komponenten verbunden werden um ein lauffähiges System zusammenzusetzen. Nicht benötigte Komponenten können hierfür weggelassen werden und ermöglichen so, ein sehr schlankes System zu entwerfen. Dies ist insbesondere in Ubicomp-Anwendungen ein großer Vorteil, da viele Geräte über beschränkte Ressourcen verfügen. Allerdings fehlen Erfahrungen aus dem praktischen Einsatz um dieses System abschließend bewerten zu können.

Literatur

1. P. Costa, G. Coulson, C. Mascolo, G. P. Picco, and S. Zachariadis. „The RUNES Middleware: A Reconfigurable Componentbased Approach to Network Embedded Systems“ In Proc. of 16th International Symposium on Personal Indoor and Mobile Radio Communications (PIMRC05). IEEE Press, Sept. 1995
2. C. Mascolo, S. Zachariadis, G. P. Picco, et. al. „RUNES Middleware Architecture“, RUNES, Nov. 2005, http://www.ist-runes.org/docs/deliverables/D5_02_01.pdf
3. Michi Henning. „A new Approach to Object-Oriented Middleware“, IEEE Internet Computing, Jan/Feb 2004, http://www.triodia.com/staff/michi/ieee/ieee.pdf
4. Michi Henning. „Distributed Programming with ICE“, ZeroC, 2003, http://www.zeroc.com/Ice-Manual.pdf
5. U. Hammerschall. „Verteilte Systeme und Anwendungen“, Pearson Studium, Mnchen, 2005
6. F. Jammes, A. Mensch, and H. Smit. „Service-Oriented device communications using the devices profile for web services“, In MPAC '05: Proceedings of the 3rd international workshop on Middleware for pervasive and ad-hoc computing, pages 1-8. ACM Press, 2005

Protokolle und Interfaces für die Backend-Frontend-Kommunikation in ubiquitären Informationsumgebungen

Götz Bürkle[1]

Telecooperation Office, Institut für Telematik, Universität Karlsruhe (TH)
SAP Research
goetz@buerkle.org

Zusammenfassung Mit der zunehmenden Miniaturisierung und der damit einher gehenden Verbreitung immer kleiner werdender „Computer“ kommen langsam neue Probleme zum Vorschein, an die vor einigen Jahren noch niemand dachte.
Heutzutage steckt in jedem Kleinstgerät ein „Rechner“, der immer häufiger auch mit seiner Umwelt kommuniziert. Als diese Geräte noch Träume waren wagte man noch nicht sich auszumalen, mit welchen Problemen man konfrontiert werden könnte, wenn die Zahl dieser Geräte stark ansteigen würde.
Jetzt, wo ubiquitäre Technologien eine gewisse Verbreitung gefunden haben und zum Beispiel als Sensornetze kein rein akademisches Thema mehr sind, muß man sich Gedanken machen, mit dem steigenden Kommunikationsaufkommen zwischen datengenerierenden Frontends und einem datenverarbeitenden Backend fertig zu werden.
Verschiedene Lösungsansätze dieser Problematik und einige beispielhafte Implementierungen derselben werden in dieser Arbeit vorgestellt.

1 Einleitung

Mit zunehmender Miniaturisierung und sonstiger Weiterentwicklung verschiedener Technologien werden die Möglichkeiten immer größer, ubiquitäre Informationsumgebungen zu etablieren. Mark Weisers Vision des „Ubiquitous Computing“, die er in „The computer for the 21st century“ [1] formulierte ist heute in manchen Bereichen zumindest in Labors schon Realität.
Wie Endres et al. in „A Survey of Software Infrastructures and Frameworks for Ubiquitous Computing“ [2] beschreiben wächst das Interesse in diesem Gebiet weiter zu forschen immer noch.

Beim Begriff „ubiquitäre Informationsumgebungen“ sollte man nicht an das denken, was man heute unter vernetzten „Personal Computern“ versteht, sondern ein System von vielen geographisch verteilten und vernetzten Geräten. Überwiegt der sensorische Aspekt spricht man oft von Sensornetzwerken.

Das Thema mit dem sich diese Arbeit beschäftigt, die Backend-Frontend-Kommunikation in ubiquitären Informationsumgebungen, wirft vor allem in Sensornetzwerken Probleme auf, auf die ich im Verlauf dieser Arbeit eingehen und Lösungsansätze vorstellen werde.

Im Folgenden Abschnitt werde ich zuerst kurz einige Anwendungsbeispiele für ubiquitäre Informationsumgebungen aus verschiedenen Bereichen des alltäglichen Lebens vorstellen, um die Wichtigkeit und praktische Bedeutung des Themas aufzuzeigen.

Danach wird das Überlastungsproblem des Backends in der Backend-Frontend-Kommunikation aufgezeigt, um später im selben Abschnitt allgemeine Lösungsansätze für diese Problematik vorzustellen.

Nachdem ich allgemeine Lösungsansätze für diese Probleme, die schwerpunktmäßig in datenbankorientierte, hierarchieorientierte und ereignisorientierte Methoden aufgeteilt sind, angeführt habe will ich einige konkrete Implementierungen dieser Ansätze beispielhaft vorstellen.

Im letzten Abschnitt werde ich versuchen ein Fazit aus dieser Arbeit zu ziehen.

2 Anwendungsbeispiele

Um einen Eindruck davon zu bekommen, um welche Art von Anwendungen es hier geht will ich zuerst einige Beispiele aufzeigen, auf die ich dann später wieder verweisen kann. Als erstes Beispiel nehme ich den DigiClip (siehe Beigl et al., „DigiClip: Activating physical documents“ [3]), als zweites gehe ich kurz auf Möglichkeiten in der Lagerverwaltung (siehe Haller und Nochta, „Kooperation zwischen intelligenten Gütern“ [4]) oder der Produktion ein und als drittes skizziere ich kurz den Einsatz eines Sensornetzwerks zur Überwachung des Straßenverkehrs (siehe beispielsweise Madden et al., „Fjording the Stream: An Architecture for Queries Over Streaming Sensor Data“ [5]).

2.1 DigiClip

Dieses System zielt darauf ab ausgedruckte Dokumente mit ihren digitalen „Originalen“ zu verknüpfen, Veränderungen anzuzeigen und durch räumliche Verfolgung der Dokumente auch festzustellen, ob die Dokumente nur an Orten bzw. in Räumen gelesen werden, an bzw. in denen dies erlaubt ist (sozusagen eine Abbildung des Rechtemanagements des Dokumentenverwaltungssystems).

2.2 Lagerverwaltung oder Produktion

Ein sehr typisches Beispiel für den Einsatz von Sensornetzen ist auch der Bereich der Lagerverwaltung oder in der Montage. In einem großen Lager werden täglich Unmengen an Waren ein- und ausgelagert. Von großem Interesse ist es möglichst effizient festzustellen, welche Waren wo eingelagert und wieder ausgelagert werden oder abzufragen, welche Waren sich überhaupt wo im Lager befinden.

Ein ähnliches Anwendungsgebiet ist die Produktion, wo es sinnvoll sein kann zu überwachen, welche Teile z. B. im Produktionsprozeß eines Automobils montiert werden um später z. B. das Produkt allein durch Integritätsbedingungen auf Vollständigkeit überprüfen zu können.

In der Literatur findet man beispielsweise in Haller und Nochta, „Kooperation zwischen intelligenten Gütern“ [4] und in Bonnet et al., „Towards Sensor Database Systems“ [6] derartige Beispiele.

2.3 Überwachung des Straßenverkehrs

Im Straßenverkehr gibt es viele Möglichkeiten mittels Sensoren Daten zu erheben. In „Fjording the Stream: An Architecture for Queries Over Streaming Sensor Data“ [5] erwähnen Madden et al. ein System, das aus sehr vielen Induktionsschleifen besteht, womit es möglich ist den Verkehrsfluß über die Anzahl, Länge und Geschwindigkeit der Autos zu ermitteln.
Natürlich könnte man am Straßenrand auch Sensoren zur Emissionsmessung installieren oder versuchen mit Hilfe von Sensoren die Straßenverhältnisse wie Nässe, Nebel oder Glätte festzustellen. Hier sind also viele Einsatzmöglichkeiten denkbar.

Bei der Überwachung des Straßenverkehrs könnte man auch, wie Li et al. in „Event Detection Services Using Data Service Middleware in Distributed Sensor Networks“ [7] ausführen, neben den Sensoren Menschen mit weiteren mobilen Geräten wie beispielsweise PDAs mit den Sensordaten versorgen und sie so bei Ihrer Arbeit unterstützen.

3 Backend-Frontend-Kommunikation

Nachdem nun klar ist, um welche Arten von Anwendungen es überhaupt geht werde ich in diesem Abschnitt auf die grundsätzliche Problematik, die solche Anwendungen mit sich bringen können, eingehen, wozu ich zu Beginn einige Begriffe definieren werde, um dann darzustellen, wie das Problem entsteht und welche Ideen zur Lösung desselben in der Literatur diskutiert werden.

3.1 Grundsätzliche Problematik

Was sind Backend und Frontend
Im Bereich der ubiquitären Informationsumgebungen versteht man unter Backend in der Regel einen „zentraler Kern“ eines Systems. Oft ist dieser „zentrale Kern“ ein Server, der bestimmte Aufgaben übernimmt (zur Definition siehe auch „RUNES: Survey of Middleware for Networked Embedded Systems“ [8]).

Im Gegensatz dazu bezeichnet man die verteilten Komponenten eines solchen Systems als Frontend. In diesem Sinne versteht man unter „Frontend“ also nicht die Benutzerschnittstelle, wie man meinen könnte, sondern die Komponenten, die die Daten generieren.

Im vorgenannten Anwendungsbeispiel DigiClip stellen die DigiClips das Frontend dar, während der zentrale Server mit dem Dokumentenmanagementsystem das Backend darstellt.

In den Beispielen zur Lagerhaltung oder Produktion aus dem letzten Abschnitt sind die Komponenten, die die eingelagerten oder ausgelagerten Waren

erfassen bzw. die Komponenten, die den Einbau von Teilen erfassen das Frontend, während das System, das diese Daten dann weiterverarbeitet, das Backend ist.

Beim Beispiel der Überwachung des Straßenverkehrs bilden die verteilten Sensoren „auf der Straße“ das Frontend, während das Backend diese Daten sammelt und auswertet.

Was passiert zwischen Backend und Frontend

Im Fall eines Sensornetzwerkes sind also die verteilten Sensorknoten die Frontend-Komponenten. Diese erheben irgendwelche Daten, doch was passiert mit den Daten, wenn sie gemessen wurden? Sie müssen irgendwohin übermittelt werden, wo sie dann weiterverarbeitet werden können. Die Weiterverarbeitung geschieht, so zumindest der einfachste Ansatz, auf einem zentralen System, dem Backend. Und diese Datenübertragung zwischen Frontend-Komponenten und dem Backend ist die sogenannte Backend-Frontend-Kommunikation.

In kleineren Systemen kann man meist keine Probleme an dieser Architektur erkennen, doch sobald man die Zahl der Frontend-Komponenten erhöht, erhöht sich auch das Kommunikationsaufkommen zwischen dem Backend-System und den Frontend-Komponenten.

Jede Frontend-Komponente ist in der Lage eine bestimmte Bandbreite auszunutzen und je mehr Frontend-Komponenten man nun in das System einbindet, desto größer wird die gesamte Bandbreite mit der die Frontend-Komponenten kommunizieren können. Das zentrale Backend-System kann zwar bis zu einem gewissen Punkt soweit ausgebaut werden, daß es das zunehmende Datenaufkommen verarbeiten kann, doch früher oder später stößt man beim Backend-System an die Grenzen des technisch Machbaren, früher wahrscheinlich schon an die Grenzen des wirtschaftlich Vertretbaren.

Das Backend-System skaliert also nicht so gut, wie die dezentralen, verteilten Frontend-Komponenten, weswegen an der Schnittstelle zwischen dem Backend-System und den Frontend-Komponenten die Gefahr einer Überlastung besteht. Genau um dieses Überlastungsproblem soll es im Folgenden gehen.

Zuerst werde ich einige grundlegenden Lösungsansätze für dieses Problem darlegen, um danach kurz auf konkrete Implementierungen dieser Ansätze einzugehen.

3.2 Ideen für Lösungsansätze

Ein relativ weit verbreiteter Ansatz besteht in einer zunehmenden „Dezentralisierung des Backend-Systems“ in das (Sensor-)Netzwerk hinein, wodurch der zentrale Teil des Backend-Systems weniger Daten verarbeiten und mit weniger Frontend-Komponenten direkt kommunizieren muß, da bereits im Netzwerk eine Datenverarbeitung stattfindet, das sogennannte „in-network processing“.

Sensornetzwerke als Datenbanken

Solche Ansätze sehen das Sensornetz oft nicht mehr nur als einfaches Netz von

Sensoren, sondern gehen teilweise so weit, das Netz an sich als Datenbank anzusehen (siehe auch Govindan et al., „The Sensor Network as a Database“ [9] und Bonnet et al., „Towards Sensor Database Systems“ [6]). Die Benutzeranfragen gehen dann nicht mehr an einen zentralen Datenbestand, der von der Backend-Komponente verwaltet wird, sondern höchstens über das Backend in das (Sensor-)Netzwerk, dessen Knoten dann Teile der Anfrage abarbeiten und dem Backend-System bereits das Ergebnis dieses Teils mitteilt, so daß auf diese Weise das Backend-System entlastet wird.
In der Literatur findet man Begriffe wie lokalisierte Algorithmen bei Estrin et al. in „Next Century Challenges: Scalable Coordination in Sensor Networks“ [10], oft wird auch ein verteilter Ansatz einem Warehousing Ansatz gegenübergestellt, wie von Bonnet et al. in „Querying the Physical World“ [11] und „Towards Sensor Database Systems“ [6], Govindan et al. in „The Sensor Network as a Database“ [9] und Beutel et al. in „Prototyping Wireless Sensor Network Applications with BTnodes“ [12].
Der Datenbestand liegt also verteilt im Netz vor, beispielsweise auch als verteilte Tupel-Räume wie Davies et al. in „Limbo: A Tuple Space Based Platform for Adaptive Mobile Applications“ [13] oder Costa et al. in „TeenyLIME: Transiently Shared Tuple Space Middleware for Wireless Sensor Networks“ [14] ausführen.

Diese Verschiebung von Funktionalität aus dem Backend in das Netzwerk bzw. in mehrere Netzwerkkomponenten ist den meisten Ansätzen ähnlich, jedoch unterscheiden diese sich in teilweise interessanten Details.

Passend zu der Sicht ein Sensornetzwerk als eine Art Datenbank aufzufassen nutzen viele Ansätze als Grundlage für die Abfrage von Informationen die Abfragesprache für relationale Datenbanken SQL (Structured Query Language) und erweitern diese dann, um sie auf die verteilte Infrastruktur anwendbar zu machen, wie es von Bonnet et al. in „Querying the Physical World“ [11] und „Towards Sensor Database Systems“ [6], Shen et al. in „Sensor Information Networking Architecture and Applications“ [15], Li et al. in „Event Detection Services Using Data Service Middleware in Distributed Sensor Networks“ [7], Madden et al. in „TinyDB: an acquisitional query processing system for sensor networks“ [16] oder Mascolo et al. in „RUNES: Survey of Middleware for Networked Embedded Systems“ [8] beschrieben wird.

Clustering - Bildung einer Hierarchie

Zentrale Bedeutung hat in den meisten Protokollen das Bilden einer, wie auch immer gearteten Hierarchie, also das Clustering. Wobei sich hier die einzelnen Ansätze unterscheiden. Einige nutzen alle oder auch nur ausgewählte Sensorknoten selbst zur Vorverarbeitung, andere wie Beutel et al. in „Prototyping Wireless Sensor Network Applications with BTnodes“ [12] schlagen vor dafür Extra-Komponenten wie PDAs als sogenannte „cluster-heads“ zu benutzen. Interessant an deren Ansatz ist insbesondere die Tatsache, daß die Software (sog. Smoblets) zur Verarbeitung nicht auf den PDAs selbst vorhanden sein muß, sondern von Sensorknoten bezogen werden kann.

In „Distributed Top-Down Hierarchy Construction“ [17] beschreiben Thaler et al. mit dem TDH (top-down hierarchy) Algorithmus eine effiziente Methode, um eine Hierarchie verfeinernd aufzubauen, die anderen aufbauenden Verfahren überlegen ist, da sie weniger Ressourcen benötigt.

Arbeiten mit Ereignissen

Weitere Ideen beschäftigen sich damit, die Kommunikation im Netz zu reduzieren und Energie zu sparen, indem man sie von Ereignissen abhängig macht oder den Kontext mit einbezieht, siehe Hill et al. in „System Architecture Directions for Networked Sensors“ [18] und Beigl et al. in „AwareCon: Situation Aware Context Communication“ [19].
Es kann sich lohnen unnötige Kommunikation zu verhindern, wenn man, wie Zhao et al. in „Information-driven dynamic sensor collaboration“ [20] vorschlagen, feststellen kann welcher Sensor zu welchem Zeitpunkt mit welchen anderen Sensoren kommunizieren soll.
In „Middleware infrastructure for context-aware ubiquitous computing systems“ [21] gehen Shehzad et al. sogar so weit die Behauptung aufzustellen, daß ereignisbasierte Middleware möglicherweise besser skaliert als andere („It is claimed that event based middleware has potentially better scaling properties for such applications than object based middleware.“).

Eine weitere Herausforderung könnte darin liegen herauszufinden, welche Daten überhaupt interessant sind, und welche überhaupt keiner weiteren Verarbeitung bedürfen. In „Event Detection Services Using Data Service Middleware in Distributed Sensor Networks“ [7] nennen Li et al. ein Beispiel, in dem man die Semantik mitnutzen kann und so bestimmte Meßdaten nicht in allen Umgebungen Alarme hervorrufen müssen.

Im Kern geht es bei allen Dezentralisierungsbestrebungen darum, daß Sensorknoten (lokal) zusammenarbeiten und so die Qualität der Daten verbessern, das Netz weniger fehleranfällig wird, die Ressourcen besser genutzt werden und vor allem die Skalierbarkeit deutlich erhöht wird, siehe Zhao et al. in „Information-driven dynamic sensor collaboration“ [20] und Hellerstein et al. in „Adaptive Query Processing: Technology in Evolution“ [22].

4 Implementierung der Ansätze/konkrete Protokolle

In diesem Abschnitt werde ich versuchen einige verschiedene Implementierungen ein wenig zu beschreiben. Die Systeme habe ich grob zeitlich angeordnet. Zuerst werde ich etwas zum COUGAR Projekt (COUGAR: The Network Is The Database, [6]) schreiben, danach zu SINA (Sensor Information Networking Architecture, [15], [8]), um dann zu Fjords (Framework in Java for Operators on Remote Data Streams, [5]) zu kommen. Als nächstes werde ich auf DSWare (Real-Time Event Detection Service using Data Service Middleware, [7], [8]) und dann auf MiLAN (Middleware Linking Applications and Networks, [23], [8]) eingehen um schließlich mit TinyDB ([16]) und RUNES (Reconfigurable Ubiquitous Networked Embedded Systems, [24]) meine Auflistung zu beenden.

4.1 COUGAR: The Network Is The Database

COUGAR wurde von Philippe Bonnet, Johannes Gehrke und Praveen Seshadri am Computer Science Department der Cornell University entwickelt.

Es setzt auf eine SQL-ähnliche Abfragesprache, wie die meisten anderen Implementierungen auch. Hervorzuheben ist bei COUGAR, daß auf jedem Knoten eine vereinfachte Version des Programms, das Anfragen abarbeitet läuft, das einfache Signalverarbeitungsfunktionen ausführt und das Ergebis zurücksendet. Der Vorteil an dieser Architektur ist, daß sie auch in großen Sensornetzen problemlos skaliert und damit auch für den Einsatz in derartigen Umgebungen geeignet ist.
Siehe auch Bonnet et al., „Towards Sensor Database Systems“ [6].

Beispielhaft wurde COUGAR mit Hilfe von Mica Motes (siehe [25]) in einem Praxistest erprobt.

4.2 SINA (Sensor Information Network Architecture)

SINA wurde von Chien-Chung Shen an der University of Delaware entworfen.

Bei diesem System wird die Position der Sensoren in den Vordergrund gestellt. Auch das hierarchische Clustering richtet sich nach der Umgebung und der Leistung der Knoten. In jedem Cluster wird ein sogenannter „cluster head“ ausgewählt, wie der Clustering-Algorithmus genau funktioniert kann man Estrin et al. „Embedding the Internet“ [26] entnehmen.
Die „cluster heads“ dienen auch als Caches und halten beispielsweise aggregierte Daten der Knoten, die im selben Cluster sind, vor. Der Vorteil besteht darin, daß bei Anfragen die Antwortzeit dadurch verkürzt wird, daß nur noch die „cluster heads“ abgefragt werden müssen, jedoch erkauft man sich diesen Zeitvorteil durch ungenauere Ergebnisse, denn die einzelnen Knoten innerhalb eines Clusters aktualisieren die Werte des „cluster heads“ nicht kontinuierlich sondern periodisch.
Die gesammelten Daten werden bei SINA zu einem inhärenten Teil des jeweiligen Sensorknotens. Wie auch bei COUGAR und MiLAN läuft bei SINA auf jedem Sensorknoten eine Ausführungsumgebung, die ankommende Nachrichten abarbeitet, diese nennt sich abgekürzt SEE („sensor executon environment“). Die Nachrichten bestehen aus zwei Teilen. Den eigentlichen Inhalt bildet eine Anfrage, die in der Sensor Query and Tasking Language (SQTL) geschrieben wird. Dies ist eine prozedurale Skriptsprache, deren Syntax von SQL abgeleitet wurde, näheres zu SQTL wird in „Querying and Tasking in Sensor Networks“ von Jaikaeo et al. [27] beschrieben. Es ist auch möglich in SQTL mit einigen Ereignissen zu arbeiten. Dieser Inhalt wird in einen sogenannten „SQTL wrapper“ (auch „header“ genannt) eingekapselt, der ein XML-Gerüst darstellt.
Bei SINA werden auch die datenzentrierten Eigenschaften von Anfragen in Sensornetzen berücksichtigt, weswegen anstatt expliziter Adressierung ein merkmalsbasierter Ansatz gewählt wurde. Um Netzwerkbandbreite zu sparen wird auf SPIN (siehe Heinzelmann et al., „Adaptive protocols for information dissemination in wireless sensor networks“ [28]) gesetzt.

Der Begriff Frontend-Knoten wird bei SINA anders gebraucht, als ich ihn gebrauche. Bei SINA versteht man darunter einen speziellen Knoten, der direkt an das Netzwerk angeschlossen ist und über den die Anfragen in das Sensornetz verteilt und verarbeitet werden. In anderen Systemen wird diese Komponente oft „Gateway-Knoten“ genannt.
Eine Besonderheit von SINA besteht darin, wie mobile Benutzer und das stationäre Netzwerk miteinander interagieren können. Die verwendete Technik, die gewährleisten soll, daß ein mobiler Benutzer über einen Netzwerkknoten Daten anfragt, sich bewegt und das Ergebnis dann von einem anderen Netzwerkknoten geliefert bekommt nennt sich „progressive footprint chaining“. Als Beispielanwendung wird die Kursverfolgung von Fahrzeugen angeführt.
Als weitere Besonderheit werden unterschiedliche Methoden Informationen zu sammeln auf unterschiedlichen Ebenen innerhalb einer Anwendung genutzt um die Gesamtleistung zu verbessern.
In „RUNES: Survey of Middleware for Networked Embedded Systems“ [8] wird von Mascolo et al. jedoch kritisch bemerkt, daß es bislang noch keine Implementierung dieser Architektur gibt.
Siehe auch Shen et al., „Sensor Information Networking Architecture and Applications“ [15].

4.3 Fjords (Framework in Java for Operators on Remote Data Streams)

Fjords wurde von Samuel Madden und Michael J. Franklin an der University of California entwickelt.

Bei Fjords wird davon ausgegangen, daß Sensoren ununterbrochen Daten liefern, also einen Datenstrom erzeugen. Weil die Datensröme unendlich sind, laufen auch die Anfragen in der Regel immer weiter. Die Sensorknoten mit ihren beschränkten Ressourcen übernehmen hier keine Verarbeitungsaufgaben sondern sammeln lediglich Daten, fassen diese zu Paketen zusammen und verschicken sie dann, gegebenenfalls über sogenannte „sensor's proxies“ an einen verarbeitenden Knoten, in der Regel ein gut angebundener Server. Sensor Proxies spielen bei Fjords eine wichtige Rolle, denn über sie kann die „Abtastrate“ der Sensoren an die Nachfrage nach diesen Daten angepaßt werden, was viel Energie einsparen kann. Außerdem können über die Sensor Proxies die Programme auf den Sensorknoten auch komplett ausgewechselt werden. Sensorknoten werden bei Fjords als objektrelationale Tabellen aufgefaßt, es wird die Implementierung benutzt, die auch für das Cougar Projekt in „Towards Sensor Database Systems“ von Bonnet et al. [6] vorgeschlagen wird. Wie in anderen Ansätzen (z. B. Bonnet et al., „Towards Sensor Database Systems“ [6]) ist auch hier eine Kombination von Sensordaten mit „statischen Quellen“ möglich.
Fjords kombiniert Proxies mit sich nicht gegenseitig blockierenden Operatoren auf Datenströmen und herkömmlichen Anfrageplänen. Das Anwendungsbeispiel zur Überwachung des Straßenverkehrs mit dem Berkely Highway Lab (BHL) habe ich Madden und Franklin, „Fjording the Stream: An Architecture for Queries

Over Streaming Sensor Data“ [5] entnommen, denn dort wird es als Testumgebung aufgeführt, anhand dessen gezeigt wird, daß es mit Hilfe der Fjords Architektur möglich ist viele Anfragen gleichzeitig abzuarbeiten. Im Gegensatz zu den meisten anderen Ansätzen wird bei Fjords keine spezielle Anfragesprache eingesetzt, denn die Konzepte sind auf die meisten Sprachen übertragbar.
Siehe auch Madden und Franklin, „Fjording the Stream: An Architecture for Queries Over Streaming Sensor Data“ [5].

4.4 DSWare (Real-Time Event Detection Service using Data Service Middleware)

DSWare wurde von Shuoqi Li, Ying Lin, Sang H. Son, John A. Stankovic und Yuan Wei am Department of Computer Science an der University of Virginia entwickelt.

Es handelt sich hierbei um eine flexible datenzentrierte Middleware, die die Konzepte eines Speicherdienstes aus „Data-centric storage in sensornets“ von Shenker et al. [29] implementiert. Es besteht die Möglichkeit miteinander in Beziehung stehende Daten örtlich benachbart zu speichern. Die „Speicherhierarchie“ ist von der phyischen Netzwerkierarchie entkoppelt, so können im Speicher-Overlay-Netzwerk mehrere physikalische Sensorknoten dem selben logischen Knoten zugeordnet werden. Im Gegensatz zu anderen Middlewares ist bei DSWare das Routing ausgelagert. Dadurch, daß bei DSWare Daten redundant gespeichert werden können, wirken sich Ausfälle einzelner Knoten solange nicht aus, wie irgendein Sensor die angeforderten Daten liefern kann (siehe Heinzelmann et al., „Middleware to support sensor network applications“ [23]). Die Replikation von Daten koordinieren die Knoten selbständig, so daß sie ihre Daten dann kopieren, wenn sie nicht ausgelastet sind. Der Vorteil mehrere Kopien von Daten im Sensornetz „gecached“, also redundant, vorliegen zu haben, die am häufigsten angefragt werden wird eventuell durch den Wartungsaufwand für die Kopien wieder aufgehoben. Hier muß man den Zielkonflikt zwischen schneller Antwortzeit und erhöhtem Wartungsaufwand betrachten. Für jegliche Art der örtlichen Zusammenarbeit verschiedener Knoten ist die Gruppen Management Komponente verantwortlich, die diese Funktionalität bereitstellt. Als typische Anfragen werden sogenannte „data subscription queries“ aufgeführt, deren praktische Relevanz an einem Beispiel illustriert wird, an das das Anwendungsbeispiel der Überwachung des Straßenverkehrs angelehnt ist. Außerdem bilden Ereignisse die Grundlage für DSWare. Um Ereignisse zu registrieren oder zu löschen wird auch hier eine SQL-ähnliche Syntax benutzt, wobei die Entscheidung bewußt für eine SQL-ähnliche Syntax gefällt wurde, obgleich dies einen gewissen Parsingoverhead mit sich bringt. Hervorzuheben ist bei DSWare, daß es eine Ereignis-Hierarchie gibt, so daß es neben atomaren Ereignissen auch zusammengesetzte Ereignisse gibt. Interessant ist auch der Ansatz, den ich weiter oben bei den allgemeinen Ideen schon kurz erwähnt habe, daß bei DSWare auch die Semantik von Daten mitbenutzt werden kann, um zum Beispiel Fehlalarme zu reduzieren. Eine Ereigniserkennung, die der Bedeutung von Kontexten Rechnung trägt, kann viel Energie sparen, was in Sensornetzen ein wichtiges Argument ist.

In den Abbildungen 4 und 5 in „Event Detection Services Using Data Service Middleware in Distributed Sensor Networks“ [7] zeigen Li et al. auch, daß die Reaktionszeit und die Netzwerkbelastung durch die Nutzung von DSWare relativ gering ist.
Durch seine Architektur bietet DSWare die Möglichkeit Daten verläßlich zu speichern, um die Echtzeit-Performance und die Verläßlichkeit von aggregierten Ergebnissen zu verbessern, außerdem wird der Kommunikationsoverhead verringert. So können, wie Mascolo et al. in „RUNES: Survey of Middleware for Networked Embedded Systems“ [8] beschreiben Sensornetze von Anwendungen über fast genau die selben Schnittstellen benutzt werden wie gewöhnliche Datenbanken.
Siehe auch Li et al., „Event Detection Services Using Data Service Middleware in Distributed Sensor Networks“ [7].

4.5 MiLAN (Middleware Linking Applications And Networks)

MiLAN wurde von Wendi B. Heinzelman, Amy L. Murphy, Hervaldo S. Carvalho und Mark A. Perillo vom Center for Future Health, Dept. of Electrical and Computer Engr. und Computer Science Department der University of Rochester entwickelt.

Sie entwarfen eine adaptive Middleware, bei der wie auch bei COUGAR auf jedem Sensorknoten eine Version von MiLAN läuft (Mascolo et al., „RUNES: Survey of Middleware for Networked Embedded Systems“ [8]). MiLAN wird auch als proaktives Middleware-System bezeichnet, weil alle Sensorknoten in die Verarbeitung miteinbezogen werden. Wie Weis in „Adaptive Systeme“ [30] so treffend formulierte koordiniert MiLAN die empfangenen Informationen und optimiert die Abläufe beziehungsweise passt die Charakteristik des Sensornetzes entsprechend an. Dies wird dadurch begünstigt, daß MiLANs Architektur bis in den Netzwerkprotokollstack hineinreicht, was für eine Middleware eher unüblich ist.
Siehe auch Heinzelmann et al., „Middleware to support sensor network applications“ [23].

4.6 TinyDB

TinyDB wurde von Samuel Madden vom Massachusetts Institute of Technology, Michael Franklin und Joseph Hellerstein von der University of California und Wei Hong von Intel Research entworfen.

Bei TinyDB handelt es sich um eine verteilte Ausführungsumgebung für Anfragen in einem Sensornetz, die auf jedem Sensorknoten läuft. TinyDB baut auf TinyOS auf. TinyDB geht über simples Verarbeiten im Netzwerk hinaus und bietet mit ACQP („acquisitional query processing“) weitergehende Möglichkeiten für Anfragen in Sensornetzen. Anfragen gelangen über einen gut ausgestatteten PC, die sogenannte basestation, in das Sensornetz. Dieser PC parst die Anfrage, optimiert sie und sendet sie dann in das Sensornetz. Die „basestation“ bildet die Wurzel des Routing-Baumes, die Sensoren senden ihre Ergebnisse einfach auf

dem entgegengesetzten Weg zurück, auf dem sie die Anfrage erhalten haben. Hervorzuheben ist bei TinyDB die Nutzung von semantischen Routing Bäumen („semantic routing tree“, SRT), bei denen auch semantische Eigenschaften beim Aufbau des Baumes berücksichtigt werden.
Nach den Erfahrungen, die Madden et al. in „TinyDB: an acquisitional query processing system for sensor networks“ [16] beschrieben haben ist der in TinyDB gewählte geclusterte SRT-Ansatz anderen SRT-Algorithmen überlegen.
SRTs können die Anzahl der Knoten, die in die Verarbeitung einer Anfrage mit einbezogen werden reduzieren. Ein Nachteil von SRTs liegt in den hohen Kosten die entstehen, wenn Sensorknoten ihre übergeordneten Knoten wechseln. Die Kosten die entstehen, um einen SRT aufzubauen sind vergleichbar mit den Kosten anderer Verfahren.
Wie schon in anderen Systemen kommt auch bei TinyDB eine SQL-ähnliche Anfragesprache zur Anwendung.
Vier Fragestellungen spielen für die Verarbeitung von Anfragen eine Rolle:
(1) Wann sollen Daten für eine bestimmte Anfrage gemessen und weitergeleitet werden?
(2) Welche Sensorknoten haben überhaupt relevante Daten für eine bestimmte Anfrage?
(3) In welcher Reihenfolge sollen verschiedene Daten gesammelt und kombiniert werden und wie sollen sie mit anderen Operationen verschachtelt werden?
(4) Ist es überhaupt sinnvoll Rechenleistung und Bandbreite für bestimmte Daten zu „verschwenden“?

Zur Leistungssteigerung können sogenannte „materialization points“ angelegt werden, das sind kleine Puffer die Ergebnisse zwischenspeichern, die in anderen Anfragen genutzt werden können. Mit dieser Technik wird auch die Funktionalität implementiert, daß gesammelte Daten zwischengespeichert werden, wenn ein Sensorknoten keine Verbindung zum Netz hat. Bei Aggregationsanfragen werden die Daten bereits im Netz verarbeitet während sie den Routing Baum „nach oben“ fließen.
Im Bereich der Ereignisse ist TinyDB erst am Anfang der Möglichkeiten, denn bisher werden Ereignisse nur lokal gemeldet, aber das Potential ereignis-basierter Anfragen, die Energie sparen können ist erkannt.
TinyDB überwacht auch die Netzauslastung und reduziert die Anzahl der Pakete, wenn die Auslastung steigt, um das Netzwerk nicht zu überlasten und optimal zu nutzen.
Bislang gibt es leider noch keine Benchmarks für Sensornetzwerk-Datenbanken, aber TinyDB kann die Ergebnisse, die dem Benutzer geliefert werden erheblich verbessern. Bei TinyDB wird viel Wert auf die oben erwähnten Fragestellungen gelegt, wann, wo und in welcher Reihenfolge welche Daten gesammelt und welche Sensorknoten miteinbezogen werden sollen. Diese Fragestellungen wurden in der Literatur bisher nicht ausführlich behandelt.
Siehe auch Madden et al., „TinyDB: an acquisitional query processing system for sensor networks“ [16].

Wie COUGAR läuft auch TinyDB auf der Mote-Plattform (siehe [25]), die in Berkely entwickelt wurde.

4.7 RUNES (Reconfigurable Ubiquitous Networked Embedded Systems)

RUNES wurde von Paolo Costa und Gian Pietro Picco von der Politecnico di Milano, Geoff Coulson von der Lancaster University und Cecilia Mascolo und Stefanos Zachariadis vom University College London entworfen.

Hervorzuheben an RUNES ist die Möglichkeit, daß sich das System dynamisch an sich verändernde Umstände anpassen kann, was in Sensornetzen eine wichtige Eigenschaft ist. Wie MiLAN reicht auch RUNES bis in die Netzwerkebene hinein, was auf RUNES aufbauenden Anwendungen eine weitere Abstraktionsschicht zur Verfügung stellt.
RUNES wurde als generische Software entwickelt, die auf vielen Geräten implementiert werden kann. Wichtig bei RUNES sind die Dienste, die die verteilte Koordination regeln. Wie bei den meisten anderen Systemen setzt auch RUNES auf Clustering.
Siehe auch Costa et al, „The RUNES Middleware: A Reconfigurable Component-based Approach to Networked Embedded Systems“[24] und Mascolo et al., „RUNES: Survey of Middleware for Networked Embedded Systems“ [8].

Das System wurde auf der IST2006 (Information Society Technologies) demonstriert (siehe [31]).

4.8 Zusammenfassung in einer Tabelle

	COUGAR	**SINA**	**Fjords**	**DSWare**	**MiLAN**	**TinyDB**	**RUNES**
Universitäten	Cornell University Comp. Science Department	University of Delaware	University of California	University of Virginia Department of Comp. Science	University of Rochester	MIT University of California Intel Research	Politecnico di Milano Lancaster University University College London
Querying (Abfragesprache)	SQL-ähnlich	SQL-ähnlich	beliebig	SQL-ähnlich	-	SQL-ähnlich	beliebig
Clustering (Hierarchiebildung)	(keine Aussage)	basierend auf Nähe und Leistung der Sensorknoten [26]	ja	Storage Overlay Network	basierend auf Nähe und Leistung der Sensorknoten	Semantik wird berücksichtigt (semantic routing trees, SRT)	ja
Ereignisse	-	es gibt einige Ereignisse	-	Ereignishierarchie (atomare und zusammen-gesetzte Ereignisse)	-	nur lokale Ereignisse (direkt am Sensor)	flexible Event Notification Komponente
Beispiele	COUGAR Projekt, Mica Motes	Kursverfolgung („progressive footprint chaining“)	Berkely Highway Lab (BHL)	Überwachung des Straßen-verkehrs		Berkely-Mote-Plattform	Demon-stration auf IST2006 [31]

5 Fazit

Die grundsätzlichen Lösungsansätze sind vielversprechend. Richtig eingesetzt besteht die Möglichkeit, daß sie das Überlastungsproblem zwischen Frontend-Komponenten und Backend-System in ubiquitären Informationsumgebungen tatsächlich beseitigen oder zumindest deutlich abschwächen können.

Bei den konkreten Ansätzen kommt es stark auf den Anwendungsfall an, welches System besser geeignet ist die vorhandenen Ressourcen optimal zu nutzen.

Generell scheinen z. B. DSWare und TinyDB die interessantesten hier aufgelisteten Systeme zu sein.

DSWare besticht durch seine Ereignishierarchie, bei TinyDB sind die semantischen Routing Bäume noch einmal besonders hervorzuheben.

Wie man den zahlreichen Literaturverweisen entnehmen kann wird auf diesem Gebiet zur Zeit viel geforscht und es gibt viele unterschiedliche Systeme und Ansätze. Welche Ansätze sich längerfristig durchsetzen werden kann man heute kaum abschätzen, vor allem kann es kaum das eine optimale System geben, da dafür die Anwendungsfelder zu vielseitig sind.

Literatur

1. Weiser, M.: The computer for the 21st century. Human-computer interaction: toward the year 2000 (1995) 933–940
2. Endres, C., Butz, A., MacWilliams, A.: A survey of software infrastructures and frameworks for ubiquitous computing. Mobile Information Systems Journal **1**(1) (2005) 41–80
3. Beigl, D., Kubach, E.: Digiclip: Activating physical documents (2004)
4. Haller, S., Nochta, Z.: Kooperation zwischen intelligenten gütern. Industrie Management - Zeitschrift für industrielle Geschäftsprozesse **22**(3) (2006) 39–41
5. Madden, S., Franklin, M.J.: Fjording the stream: An architecture for queries over streaming sensor data. In: ICDE. (2002)
6. Bonnet, P., Gehrke, J., Seshadri, P.: Towards sensor database systems. Lecture Notes in Computer Science **1987** (2001) 3–??
7. Li, S., Son, S., Stankovic, J.: Event detection services using data service middleware in distributed sensor networks (2003)
8. C. Mascolo, S. Hailes; L. Lymberopoulos; G. Picco; P. Costa; G. Blair; P. Okanda; T. Sivaharan, W.F.M.K.M.R.K.F., Boulis, A.: Runes: Survey of middleware for networked embedded systems (2005)
9. Govindan, R., Hellerstein, J., Hong, W., Madden, S., Franklin, M., Shenker, S.: The sensor network as a database (2002)
10. Estrin, D., Govindan, R., Heidemann, J.S., Kumar, S.: Next century challenges: Scalable coordination in sensor networks. In: Mobile Computing and Networking. (1999) 263–270
11. Bonnet, P., Gehrke, J., Seshadri, P.: Querying the physical world (2000)
12. Beutel, J., Kasten, O., Mattern, F., Römer, K., Siegemund, F., Thiele, L.: Prototyping wireless sensor network applications with BTnodes. In: Proc. 1st European Workshop on Sensor Networks (EWSN 2004). Volume 2920 of Lecture Notes in Computer Science., Springer, Berlin (2004) 323–338

13. Davies, N., Wade, S., Friday, A., Blair, G.: Limbo: A Tuple Space Based Platform for Adaptive Mobile Applications. In: Proceedings of the International Conference on Open Distributed Processing/Distributed Platforms (ICODP/ICDP '97), Toronto, Canada (1997) 291–302
14. Costa, P., Mottola, L., Murphy, A.L., Picco, G.P.: TeenyLIME: Transiently Shared Tuple Space Middleware for Wireless Sensor Networks. In: Proceedings of the 1^{st} International Workshop on Middleware for Wireless Sensor Networks (MidSens 2006), Melbourne, Australia, ACM Press (2006)
15. Shen, C.C., Srisathapornphat, C., Jaikaeo, C.: Sensor Information Networking Architecture and Applications. IEEE Personel Communication Magazine **8**(4) (2001) 52–59
16. Madden, S.R., Franklin, M.J., Hellerstein, J.M., Hong, W.: Tinydb: an acquisitional query processing system for sensor networks. ACM Trans. Database Syst. **30**(1) (2005) 122–173
17. Thaler, D., Ravishankar, C.V.: Distributed top-down hierarchy construction. In: INFOCOM (2). (1998) 693–701
18. Hill, J., Szewczyk, R., Woo, A., Hollar, S., Culler, D.E., Pister, K.S.J.: System architecture directions for networked sensors. In: Architectural Support for Programming Languages and Operating Systems. (2000) 93–104
19. Beigl, M., Krohn, A., Zimmer, T., Decker, C., Robinson, P.: Awarecon: Situation aware context communication (2003)
20. Zhao, F., Shin, J., Reich, J.: Information-driven dynamic sensor collaboration. IEEE Signal Processing Magazine **19**(2) (2002) 61–72
21. Anjum Shehzad, Hung N. Q., K.A.P.M., Riaz, M., Liaquat, S., Lee, S., Lee, Y.K.: Middleware infrastructure for context-aware ubiquitous computing systems. ... (2005)
22. Hellerstein, J.M., Franklin, M.J., Chandrasekaran, S., Deshpande, A., Hildrum, K., Madden, S., Raman, V., Shah, M.A.: Adaptive query processing: Technology in evolution. IEEE Data Engineering Bulletin **23**(2) (2000) 7–18
23. Heinzelman, W., Murphy, A., Carvalho, H., Perillo, M.: Middleware to support sensor network applications (2004)
24. Costa, P., Coulson, G., Mascolo, C., Picco, G.P., Zachariadis, S.: The runes middleware: A reconfigurable component-based approach to networked embedded systems. In: Proceedings of the 16^{th} Annual IEEE International Symposium on Personal Indoor and Mobile Radio Communications (PIMRC'05), Berlin (Germany) (2005)
25. Ganapathi Kamath: Mica motes. http://web.syr.edu/ gkamathh/topics/micamote.html (2003)
26. Estrin, D., Govindan, R., Heidemann, J.: Embedding the Internet. Communications of the ACM **43**(5) (2000) 39–41 (special issue guest editors).
27. Jaikaeo, C., Srisathapornphat, C., Shen, C.C.: Querying and Tasking in Sensor Networks. In: SPIE's 14th Annual International Symposium on Aerospace/Defense Sensing, Simulation, and Control (Digitization of the Battlespace V), Orlando, Florida (2000)
28. Heinzelman, W.R., Kulik, J., Balakrishnan, H.: Adaptive protocols for information dissemination in wireless sensor networks. In: MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking, New York, NY, USA, ACM Press (1999) 174–185
29. Shenker, S., Ratnasamy, S., Karp, B., Govindan, R., Estrin, D.: Data-centric storage in sensornets. SIGCOMM Comput. Commun. Rev. **33**(1) (2003) 137–142

30. Weis, C.: Adaptive systeme. In: Seminar Verteilte Informationssysteme - Datenverarbeitung in Sensornetzen, Dr. Serge Shumilov, Universität Bonn, Institut für Informatik III (2006)
31. RUNES Consortium: Runes demonstration @ ist2006. http://www.ist-runes.org/news2006-11-23.html (2006)

Fault Tolerance in Wireless Sensor Networks and Robotics

Praharshana Perera

Supervisor: Luciana Moreira Sá de Souza, SAP Research

Abstract. Wireless sensor networks and robots are exposed to potentially hostile environments which can lead to failures in the system. Maintaining such systems while operating can be very costly or even impossible. Therefore, there is an increasing interest and activity in the area of reliability and fault tolerance in wireless sensor networks and robotics. In this survey, some available fault tolerance techniques in both fields are analyzed and a classification of the investigated techniques is presented.

1 Introduction

Fault tolerance is the survival attribute of computer architectures. When a system is able to recover automatically from fault-caused errors, and to eliminate faults without suffering an externally perceivable failure, the system is said to be fault tolerant[16]. It is stated in [17] that being fault tolerant is strongly related to what are called dependable systems that covers the basic requirements: availability, reliability, safety, and maintainability. Availability defines that a system is ready to be used immediately. Reliability of a system is that it can run continuously without a failure. Safety defines that when a system temporary fails to operate, nothing catastrophic happens and finally maintainability is how easy a failed system can be repaired.

Answering to the question, is it a fault, an error, or a failure? [16] says a failure occurs when a system cannot meet its requirements, an error is an undesired state that may lead to a failure, and a fault is the identified cause of the error or failure. Generally a fault tolerance system is realized by including redundant resources and implementations of fault detection and recovery algorithms in both hardware and software components. Redundancy can be achieved in hardware, in software, and in application level by redundant implementation of components. However, [16] says that design faults which are introduced by human mistakes are reproduced even when redundant copies are made.

In the past few decades fault tolerance has been addressed in distributed systems, robotics, wireless sensor networks, aeronautics, and in many other areas in computer science. These systems are real time safety critical systems that are often exposed to hazardous environments and need to be adapted to the harsh conditions even when failures are present. In this survey, we mainly focus on fault tolerance techniques in wireless sensor networks. Later, we extend our survey by looking at some available fault tolerance techniques in robotics.

1.1 Wireless Sensor Networks

A wireless sensor network is system of small, wirelessly communicating nodes where each node is equipped with multiple components [8]. Each node has a microprocessor, communication and storage subsystems, a battery supply, and sensing. Fig. 1 shows the general architecture of wireless sensor networks. [18] says that currently, networks with hundreds of such nodes are possible, but future networks should be capable of greater numbers than this, as the nodes get smaller and power efficient.

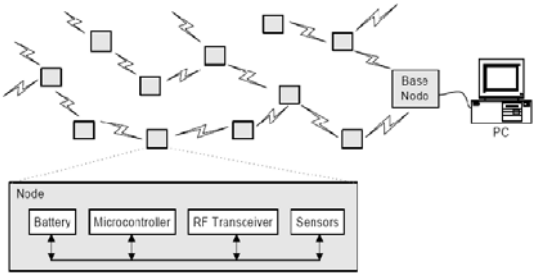


Fig. 1. Wireless sensor network architecture

Sensor based network mostly operate in potentially hostile, or at least harsh environments. For example, they can be used to sense concentrations of some chemical in an operational environment. Also they can be used in security critical applications like intelligent security systems in buildings. It is stated in [8] that sensor networks have continuous mode of operation, higher structural complexity, and sensors that have significantly higher fault rates. Additionally, maintaining and replacing components in such a network are often not possible and very expensive. Therefore, a wide amount of research has been done during the past few years to enhance fault tolerance in wireless sensor networks.

The techniques we have investigated use different fault tolerance approaches based on sensor replication. The two algorithms introduced in [1] and [2] determine a reliable sensor reading from a set of faulty sensors. The solution in [1] is based on interval graphs and [2] is an extension of [1] concerning fault detection as well. The next technique we analyze focuses on fault tolerant determination of event regions in an environment. Here, [3] and [4] propose a solution to the problem based on a Bayesian event detection scheme. [3] proposes a fault recognition algorithm and [4] propose a fault tolerance detection algorithm. Finally, in [6],[7], and [8] a fault tolerance technique based on heterogeneous back-up schemes is given. Here, they propose a method to back up one type of sensor from another to tolerate sensor failures in both binary and multi valued sensors.

1.2 Fault Tolerance in Robotics

As in wireless sensor networks, robots are often used in inaccessible or hazardous environments in order to reduce the cost and risk involved in preparing humans to operate under these conditions. For example, robots in mission critical applications, such as space and nuclear environments, must be able to operate during component failure to complete important tasks [12].

Generally, a robot is a mechanical structure consisting of links connected by joints which typically move around one axis, i.e. one degree of freedom [12]. Most robots have six degrees of freedom to position the robot in the three dimensional space. A component in the robot called the controller instructs the robot to move through its workspace based on a calculated trajectory. As shown in Fig. 2 sensors provide the information to the controller about the current position or velocity of the robot joints.

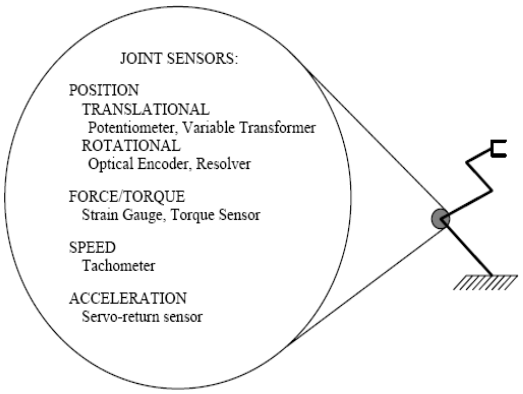


Fig. 2. Types of sensors in a robot

The robot joints are driven by an actuator, mostly actuators are electronic motors. The controller computes necessary torque to apply for each joint based on the information given by the sensors. Therefore, sensor or actuator failures provide incorrect information to the robot controller leading to a failure in the robot. But in a mission critical application the robot should function even when failures are present. Therefore, fault tolerance in robot systems has been an active research area in the past few years. In our survey, first, we introduce briefly Fault Tree Analysis (FTA) [13] in which failure paths are identified by using a graphical representation of the flow of fault events. In [10], fault tolerance capabilities of a small robot based on adaptive agents are presented. Finally, in [14], multi-layer intelligent control framework for fault tolerance in robots is presented.

1.3 Structure of the Report

In the next section, we discuss fault tolerance techniques in wireless sensor networks. Here, we mainly focus on three different techniques that tolerate failures based on replicating sensors. Next, we introduce some fault tolerance techniques in robotics. Finally, we conclude our survey with a classification of the investigated techniques.

2 Fault Tolerance techniques in Wireless Sensor Networks

In this section we address fault tolerance techniques that are employed in wireless sensor networks. Generally we identify that replication or replicating sensors is the main approach to fault tolerance. But Research in sensor replication has been addressed in various ways in wireless sensor networks. Therefore, our study will focus on different fault tolerance techniques that are based on sensor replication. First, we will discuss the techniques for tolerating failures in multi sensors, even when some of the sensors are faulty. Next, we will introduce another similar approach in tolerating faults in event region detection and finally, we will discuss heterogeneous back up schemes, where one type of sensor is backed up with another and multi-level sensor fusion, where a sensor can have multi level output value.

2.1 Fault Tolerance in Multi Sensors

Replicating sensors is desirable not only to tolerate sensor failures, but to increase the average accuracy of the ensemble of replicated sensors beyond that obtainable with a single sensor [2]. This type of replications can be applied in a multi-sensor environment or in a distributed-sensor network for the purpose of fault tolerance.

The output of a sensor can be faulty because of sensor failure or because of environment noise. A multi-sensor system has several sensors which are controlled by one processor (some times more). The main task of this approach is to get the accurate physical value of interest by combining the replicated sensor outputs. Fault tolerance in multi-sensor networks where at most *f* sensors can be faulty from *n* sensors can be summarized as:

1. How can a processor construct an estimate of a physical variable of interest *P*
2. What properties of this estimate of *P* can be exploited so that both fault-tolerance and fault-detection can be efficient.

Here, the sensors are assumed to be continues valued and the communication and synchronization issues are not taken into consideration.

2.1.1Background and Definitions

The approach defined in [2] distinguishes between three kinds of sensors as shown in Fig. 3 .An explanation of each is given below:

1. Concrete sensor: A physical device, samples the physical variable of interest
2. An abstract Sensor (ABSR): A continues function from the physical state variable to an interval of real numbers $[l_a, u_a]$ indicating the possible values that the physical variable could take. [1] says that a failure of an ABSR can arise when the underlying concrete sensor fails
3. A reliable abstract sensor (Reliable ABSR): Accurate estimate of the physical variable of interest. Contains an interval or a set of interval

In [2] the following definitions are given:

1. The width of an ABSR *A* is $(u_a - l_a)$
2. A correct ABSR *A* is one whose interval contains the actual physical value of interest and whose width is at most *T*.
3. *A* is more accurate than *B,* if $u_a - l_a < u_b - l_b$ and a correct ABSR is always more accurate than an incorrect ABSR

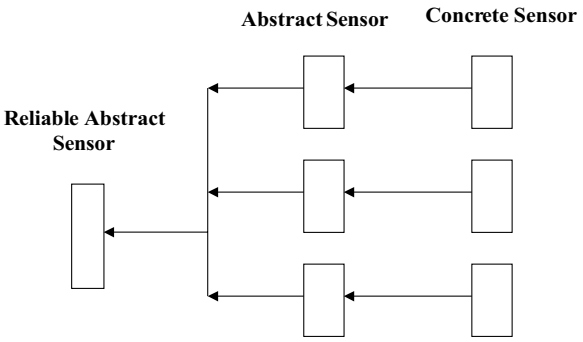


Fig. 3. Replicated Sensors

4. Given *n* intervals, and 2 <= *i* <= *n*; a distinct *i*-interval intersection is an interval in which no more than *i* intervals intersect.
5. Given intervals *A* and *B*, *A* completely overlaps *B* if $l_a < l_b$ and $u_a < u_b$
6. A connected interval sequence is a sequence which has no overlaps or gaps

In order to estimate a physical variable of interest using multiple sensors when some of them can be faulty, [1] states the following assumptions.

1. There are *n* independent ABSRs and no more that *f* are faulty
2. There is a synchronization protocol, which prevents sensor measurements from mixing up
3. If *f'* < *f* sensors have width > *T*, they can be detected a priori as faulty

Here, it can be observed that any intersection of at least (*n-f*) intervals can contain the correct value and any intersection of less than (*n-f*) intervals does not contain the correct value. Therefore, [2] suggests that a reliable ABSR, whose width < *T* can be constructed as a:

1. Single correct interval of bounded width or
2. Collection of (*n-i*) interval intersections, one of which contains the correct value

The single correct interval can be obtained by taking the union of all the intervals that are in the collection. [2] says that this technique is preferred when the communication requirements need to be kept down. However, by taking the union the points which do not belong to any possible correct interval could be in the correct interval. The collection method does not have this problem and it is said to be more accurate than the single correct interval. [2] says that ABSR as a collection of intervals has two advantages:

1. If the ABSR has to be integrated with others, the accuracy of the integrated ABSR is better than if it was single
2. If an ABSR is known to be correct and was not available at the time of sampling the physical value, then some interval in the reliable ABSR could be discarded and it improves the accuracy

Next, we will introduce two algorithms M-FTA [1] and J/FTA [2] which determine a reliable ABSR from a set of *n* intervals with at most *f* faults.

2.1.2M-FTA Technique

The M-FTA uses interval graphs to find a reliable ABSR. Each interval in a interval graph is represented with a vertex and two vertices are joined if and only if the corresponding intervals intersect. [1] defines a clique to be a set of nodes in the graph where every pair of the nodes in the set is connected by an edge. The size of a clique is the number of nodes in the set. As shown in Fig. 4 for a set of intervals n=5 with number of fault intervals f = 2. The corresponding interval graph is shown on the left. M-FTA finds all cliques of size (*n-f*) in the interval graph and each (*n-f*) clique represents an (*n-f*) interval intersection. As stated above any intersection of at least (*n-f*) intervals can contain the correct value, the cover of all cliques of size (*n-f*) contains the correct value. Therefore, as shown in Fig. 4, M-FTA provides the segment R as the single correct interval.

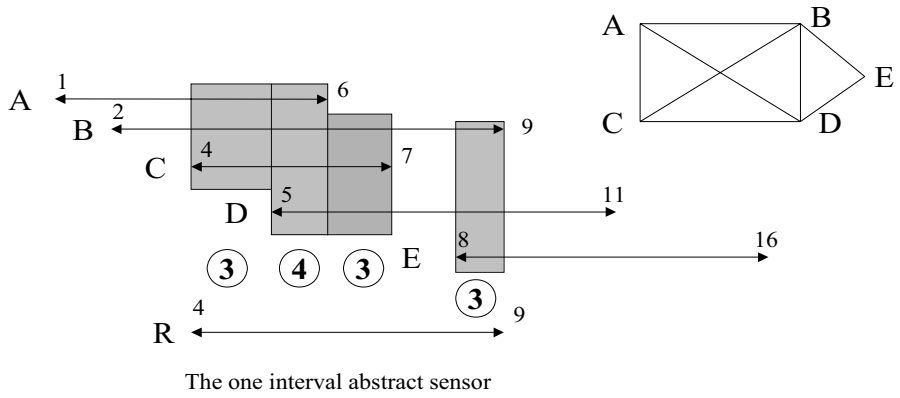


Fig. 4. Set of intervals with $n = 5$ and $f = 2$. The interval graph is shown on the left.

2.1.3J/FTA Technique

The J/FTA recognizes distinct cliques of minimum size(n-j) by sorting the n intervals according to their lower bounds $P_1, P_2, \ldots P_n$. Since at least (n-f) intervals have to intersect for the correct value to be contained in the intersection, it begins with the $(n\text{-}f)^{th}$ sorted interval $P_{n\text{-}f}$. Check if $P_{n\text{-}f}$ intersects with all the (n-f-1) earlier intervals, $P_1, P_2, \ldots P_{n\text{-}f\text{-}1}$. If it does , then record this intersecting interval, N with the label (n-f) in a set I. Like that it proceeds to the next interval $P_{n\text{-}f+1}$ and check if it intersects with the (n-f-1) earlier intervals. Record the intersecting intervals and check the set I to see if the new intersecting-interval intersects with other intervals in I. The algorithm terminates when the last interval is examined.

In the example shown in Fig. 4 J/FTA recognizes the 3 cliques ABC, BCD, BDE of size 3 and the clique ABCD of size 4 but on the other hand M-FTA recognizes all the cliques of size 3 i.e. ABC, ABD, ACD, BCD, and BDE and provides segment R as the single correct interval. J/FTA on the other hand provides the same single correct interval R and all the (n-i) $0 < i < f$ interval intersections and associates each interval I with a set of possible faulty sensors that are not part of the interval in I. For example, in Fig. 4, J/FTA associate the set of faulty sensors, assuming that there are at most two faulty sensors i.e., f=2.

1. Sensors (D,E) with the interval [4,5]
2. Sensor E with the interval [5,6]
3. Sensors (A,E) with the interval [6,7]
4. Sensors (A,C) with the interval [8,9]

Then, it allocates the set of possible faulty sensors as:

- Set of 2 faults = {D,E} or {A,E} or {A,C}
- Set of 1 faults= {E}
- Set of 0 faults = ∅

Depending on these information, J/FTA makes several inferences about faulty sensors:

1. Sensor B is not faulty
2. If A,C, or D is faulty, then another sensor must be faulty
3. Since there is a interval intersection of four sensors(Interval[5,6]) the minimum number of faulty sensors is 1(n-4, where n = 5).

[2] says that providing information about sets of possible faulty sensors is more useful than providing a list of all the possible faulty sensors.

2.2 Fault Tolerant Event Detection in Wireless Sensor Networks

This discussion focus on fault tolerant determination of event regions in an environment. An important application in sensor networks is monitoring event regions in inaccessible environments. An example of such an application from [3] is a wireless network of sensors placed in an operational environment that are capable of sensing concentrations of some chemical. The task of this network is to identify regions in the environment that contain interesting events. For example, to extract region of the network with higher concentrations of the chemical.

As stated in [4] there are two fundamental challenges in the event detection problem of a sensor network.

1. The detection accuracy is limited by the amount of noise associated with the measurement and the reliability of sensor nodes. For example, a faulty sensor may detect high chemical concentration although it is not in an event region
2. The source of energy for a sensor node is mostly an attached battery cell. A centralized event detection algorithm is not suitable in this case, since all sensors should transmit their individual sensor measurements to a central node. Therefore, a distributed algorithm is preferred in this case

A distributed solution to this problem as defined in [3] and [4], is to have each of the independent sensors make a local decision (binary) and then combine these decisions at a fusion sensor to generate a global decision. This is modeled as a hypothesis test problem, where n sensors observe an unknown hypothesis. Technically it can be explained as each sensor transmits its decision over a multiple access channel to a fusion sensor and based on these decisions the fusion sensor makes the final decision. [4] assumes that the sensor observations are statistically independent and identically distributed. [3] says that the sensor observations are spatially correlated but sensor faults are more likely to be uncorrelated.

2.2.1Background and Definitions

N sensor nodes are deployed over an interested region for the purpose of event detection. An assumption is made that the entire region is covered by the sensor network and the sensors are fully connected. It is also assumed that a sensor node can locate its neighbors within its communication range. Fig. 5 from [4] shows a sample scenario. The decisions of sensor nodes are binary (event of no-event) and independent based on its own measurements in the noisy environment. Some of the sensors can be faulty i.e. sensor reports no-event though it is in an event region or the other way. For each sensor the other sensors in the event region or in the boundary are its neighbors. [3] assumes that if sensor *i* is in the event region so are its neighbors and vice versa. It is assumed that the sensors in the boundary are relatively small.

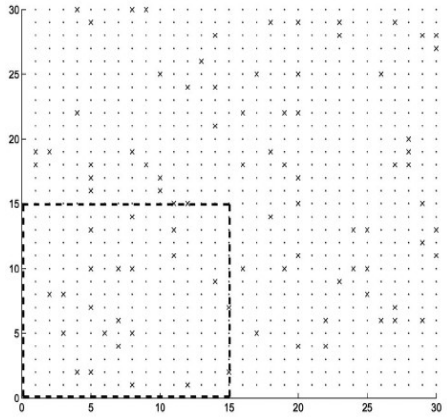


Fig. 5. A sample scenario: 900 sensors are distributed over an interested region with the single event region (enclosed inside the square with the bold dashed line). A dot denotes a healthy sensor and a cross denotes a faulty sensor.

2.2.2Distributed Event Detection

A distributed event detection scheme with n sensors and a fusion sensor is shown in Fig. 6. Given the unknown hypothesis (H_0 or H_1), each sensor makes its own decision. These decisions are combined in the fusion sensor for the final decision.

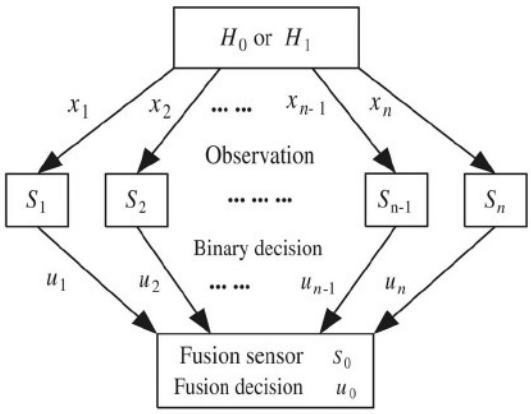


Fig. 6. Distributed Event Detection Scheme

Based on a likelihood test [5] each sensor transmits their decisions to the fusion sensor. Let u_i denote the binary decision of each sensor. Based on these received decisions the fusion sensor makes the final decision u_0. It is in the form of a *k*-out-of-*n* rule. If $u_0 = 0$ the fusion sensor decides H_0 and if $u_0 = 1$ the fusion sensor decides H_1. This can be expressed in the form:

$$u_0 = \begin{cases} 1, u_1 + \cdots + u_n \geq k \\ 0, u_1 + \cdots + u_n < k \end{cases} \quad (0)$$

Where *k* is an integer between 1 and *n*. Based on u_0 we will now discuss fault detection and correction methods that are available.

2.2.3Fault Recognition

[3] introduces a fault recognition algorithm considering the sensor fault problem. It tries to improve the sensor faults based on examining the correlation in the reading of sensors in the neighborhood. They map the real situation at the sensor T_i(H_0 or H_1) to an abstract variable S_i. The sensor reading is faulty if $T_i \neq S_i$. [3] uses a Bayesian fault detection algorithm to determine an estimate of the true reading T_i of the sensor based on the sensor readings of neighboring sensors. An assumption is made that the sensor fault probability *p* is uncorrelated and symmetric:

$$P(S_i=0 \mid T_i= H_1) = P(S_i=1 \mid T_i= H_0) \quad (0)$$

The binary sensor outputs are based on a threshold on the real-valued readings of sensors. The threshold used by [3] is based on the mean normal reading m_n and the mean event reading m_f of a sensor. They use the measure $0.5(m_n + m_f)$ as the threshold. If the errors due to sensor faults can be modeled by Gaussian distributions with mean 0 and a standard deviation σ, [3] says that the faulty probability *p* can be evaluated using the tail probability of a Gaussian, the Q-function as follows:

$$P = Q\left(\frac{(0.5(m_f + m_n) - m_n)}{\sigma} \right) = Q\left(\frac{m_f - m_n}{2\sigma} \right) \quad (0)$$

Since it also takes the readings of the neighbors into consideration, [3] defines evidence $E_i(a,k)$ such that k of the neighboring *N* neighbors of sensor *i* report the same binary reading *a* as node *i*. The estimate R_i of the true reading T_i of a sensor based on the evidence is given by:

$$P(R_i = a \mid E_i(a,k)) = \frac{k}{N} \quad (0)$$

Each sensor determines a value for R_i given information about its own sensor reading S_i and the evidance $E_i(a,k)$. The probability of estimate R_i =a given Si = b (for the two cases $b = a$ or $b = \neg a$) and $E_i(a,k)$ can be calculated using the Bayes theorem as:

$$P(R_i = a \mid S_i = b, E_i = (a,k)) = \frac{P(R_i = a, S_i = b \mid E_i(a,k))}{P(S_i = b \mid E_i(a,k))} \qquad (0)$$

Depending on the estimate for R_i a sensor can make a decision about whether or not to disregard its own sensor reading [3]. This can be done based on a threshold, i.e. if $P(R_i=a|S_i=b, E_i=(a,k))$ is above the given threshold, then Ri is set to a and the sensor reading is correct and otherwise the sensor decides its sensor reading is faulty.

2.2.4Fault Tolerant Detection

As we have already discussed, [3] considers only the sensor fault problem or it determines only if a sensor reading is false or not using a threshold. [4] says the measurement inaccuracy has direct impact on the effect fault correction and to achieve a better detection accuracy and fault tolerance, a decision scheme should take both measurement error and sensor fault into consideration. This is done by optimizing the threshold pair λ and k in section 2.2.2. [5] adjust both λ and k with sensor faults to achieve the optimal detection.

[4] defines P_F and P_D to be identical false alarm probability i.e. $P_F = P(u_i = 1|H_0)$ and identical detection probability $P_D = P(u_i = 1|H_1)$. Therefore, the system false alarm probability Q_F and the system detection probability Q_D can be given by a binomial distribution:

$$Q_F = \sum_{i=k}^{n} \binom{n}{i} P_F^i (1 - P_F)^{n-i} \qquad (0)$$

$$Q_D = \sum_{i=k}^{n} \binom{n}{i} P_d^i (1 - P_d)^{n-i} \qquad (0)$$

When q_0 and q_1 denotes the prior probabilities of
$H0$ and $H1$, under the Bayesian detection framework, the probability of detection error with n sensor nodes is given by:

$$P_e^n = q_0 Q_F + q_1 (1 - Q_D) \qquad (0)$$

[5] says that the probability of detection error is a function of λ and k and has a single minimum. The optimal λ and k that minimizes the function can be achieved by an optimization algorithm. This analysis does not consider sensor fault probability. In [4] an enhancement of this method is given considering the sensor fault probabilities.

2.3 Heterogeneous Back-Up schemes and Multi-Level Sensor Fusion in Wireless Sensor Networks

In this section we focus on heterogeneous fault tolerance techniques, where a single type of resource backs up different types of resources. The basic idea behind this technique is to adapt application algorithms to function further with the available hardware and the application needs. It is stated in [6], [7], and [8] that each of the primary types of resources: computing, storage, communication, sensing and actuating can replace each other with suitable change in application software. For example, if communication bandwidth is reduced and all of the computation power is available, the system can compress data using more computationally intensive compression schemes. The research in [6] focuses only on how to back-up one type of sensor with another since sensing has the highest fault rates. Here, we focus on multi-modal sensor fusion (MSF), a successful and a widespread heterogeneous technique that is used in embedded sensor networks (ESN). First, we illustrate multi-modal sensor fusion for binary sensors. Finally, we also discuss multi-modal sensor fusion for multi-level sensors, where sensors have multi-level output value.

2.3.1Heterogeneous Back-Up Schemes for Binary Sensors

[6], [7], and [8] informally illustrate fault-tolerant multi-modal sensor fusion for digital binary sensors using an example as shown in Fig. 7. Here, a set of objects G = {A, B, C, D, E}, which have two different attributes, X and Y are observed by sensors of types x and y. It is assumed that each object is unique, thus no two objects have identical values for both, X and Y. The attributes X and Y are used to identify each object. The objects from G are mapped into a 2-dimensional space [7]. Since binary sensors have been used, the output of a sensor is set to 0, if the objects X (or Y) attribute is less than x_i (or y_i) and 1 otherwise. A faulty sensor is represented as one stuck on a value. The sensor outputs x_i and y_i are shown by the lines X= x_i and Y= y_i in Fig. 7.

In [6] the MSF problem of uniquely identifying each object in the 2-dimensional space using minimum number of sensors is defined as selecting the minimum number of lines X= x_i and Y= y_i such that each object is within a unique grid unit formed by the lines. Two such solutions are showed in Fig. 7 **a** and **b**. Here, the objects are uniquely identified using less than the available number of sensors and by replacing each other. In **a**, the solution uses two X sensors and one Y sensor and in **b**, the solution is obtained by replacing one X sensor with an additional Y sensor. Therefore, [6] suggests a heterogeneous back up scheme for the binary multi-modal sensor fusion, where sensors of different types replace each other and cover each other faults.

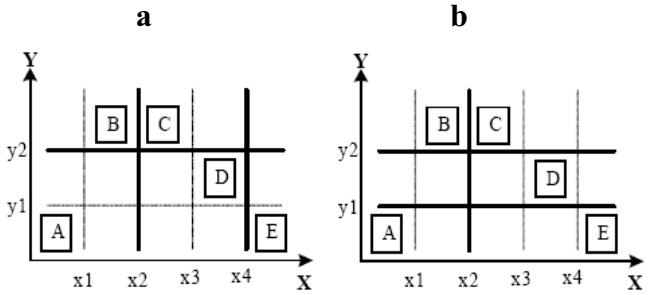


Fig. 7. An example of the MSF for binary sensors

In [6], [7], and [8] a formal definition of the multi-modal sensor allocation problem is given and an integer linear programming (ILP) formulation and an efficient heuristic is proposed to solve it. Next, we will briefly discuss the multi-modal sensor allocation problem and the proposed solution.

2.3.2Multi-modal Sensor Allocation Problem

As stated in [6], the multi-modal sensor allocation problem can be defined as: Set A of points p_i (x_{i1},…,x_{im}), in m-dimensional space where 1≤ i≤n, a positive integer J, set H that consists of $m(n-1)$ [m-1]-dimensional hyper planes, each of which is perpendicular to one of the m axes. Each hyper plane is separating two points p_i and p_j that have the closest coordinates along the particular axis to which the hyper plane is perpendicular. The goal is to find a subset of hyper planes H, such that any two points p_i and p_j are separated by at least one of the selected hyper planes and the cardinality of H is at most J.

For our discussion we summarize the multi-modal sensor allocation problem as shown in Fig. 8. In Fig. 8 **a,** we show three points *p1*, *p2*, and *p3* in the 3-dimensional space with the $m(n-1)$=3.2=6 2-dimensional hyper planes (h_1…h_6), which are perpendicular to one of the m axes. On the other hand, Fig. 8 **b** shows a possible solution to the problem. Here, the any two points are separated by at least one of the two hyper planes h_3 and h_6.

In [6], [7], and [8] two different techniques are developed to solve the allocation problem: ILP-based and simulated annealing based. The ILP-based solution defines a possible set tests (hyper planes) and minimize the total cost of all the selected tests. The simulated annealing based solution is based on four components: moves, objective function, cooling schedule, and stopping criteria[7]. Here, move is the replacement of one sensor with another and the goal is to maximize an objective function with a cooling schedule and a stopping criteria. [6], [7], and [8] say that although ILP solvers are often not fast, they are attractive since they guarantee optimal solution. In situations where ILP is not applicable the authors in [6], [7], and [8] suggest the option of using simulated annealing as the solution.

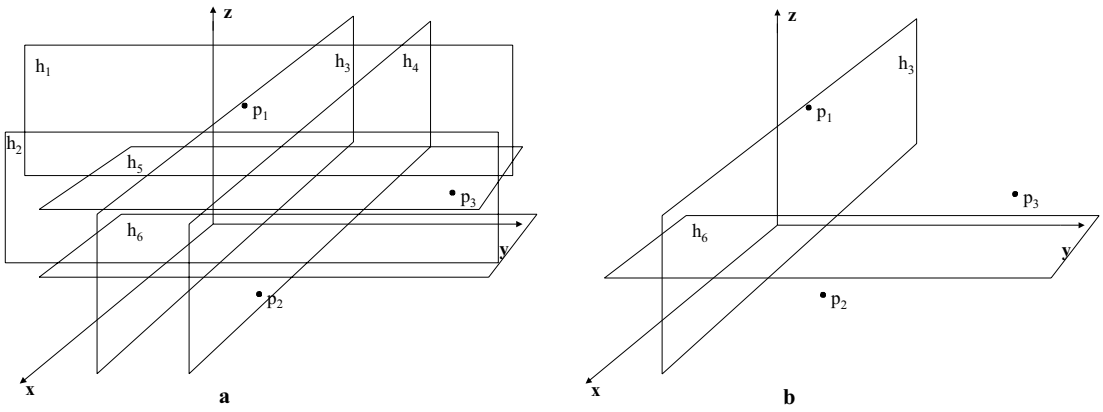


Fig. 8. Multi-modal sensor allocation problem and a possible solution

For fault-tolerance, three different mechanisms have been proposed in [8]. The first mechanism proposes that each two points have to be separated by at least *r* hyper planes. This is not acceptable, since it doubles the redundancy. The second alternative is to add exactly one extra sensor of each type to the solution generated by the sensor resource allocation problem [8]. This is an acceptable solution when a large number of sensor nodes of each type are used and a moderate level of fault-tolerance is required. [8] says that the best and the most attractive alternative is to employ a heterogeneous back up of sensors of different modality. Here, for each type of sensors for all allocations from 1 to smaller than the number allocated in the best resource allocation the cost of overall solution is calculated. Afterwards, the cost of all these solutions on y-axis and number of allocated sensors in the x-axis are plotted. Then the heterogeneous back up scheme is formed only by the allocations that are worse in terms of cost than the optimal solution and better than the solution from the second alternative [8].

2.3.3MSF for Multi-Level Sensors

In the previous section, we focused on MSF for sensors with binary output. Now, we focus on sensors that have multi-level output value. As stated in [6], a faulty sensor can have erroneous value that is varying and at a different level, compared to the actual value. In [6], [7], and [8] a system of erroneous sensor readings is considered of multi-level sensor fusion, where the sensors are of different types. As the solution to this problem, [6] suggests a known model that defines the correlations between different sensor measurements. The model can be either statistical or analytical. In the statistical modeling approach, [7] suggests a statistical leaning process to find the correlation between the sensor readings. In [6] an analytical modeling approach is proposed using equations as correlating constraint between the sensor outputs and our discussion in this section is based on the analytical modeling approach suggested in [6].

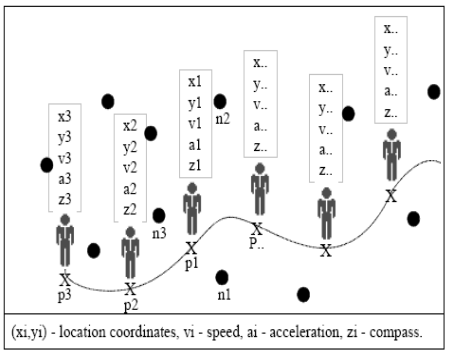


Fig. 9. Multi-level, multi-modal sensor fusion used in tracking

As introduced in [6], [7], and [8] we describe the MSF approach for multilevel sensors using a small example. As shown in Fig. 9, in a sensor network that consists of a number of nodes represented by black dots, an object *O*, a person moves along its trajectory that includes points p_i $\{1 \le i \le k\}$. It is assumed that there are four types of sensors: distance discovery, speedometer, accelerometer, and compass, which measures the angle in a 2-dimensional physical space. [7] says that three distance measurements can be used to locate the object *O* in any particular moment and Euclidian space, Newton mechanics, and trigonometry laws can be used to establish

relationships between measurements of different modalities. It is assumed that the person could communicate to the sensors that are within its communication range. The equations 9-11 can be obtained for the point $p_1(x_1,y_1)$ where O can communicate to three nodes $n_1(s_1,t_1)$, $n_2(s_2,t_2)$, and $n_3(s_3,t_3)$. Here, R_1, R_2, and R_3 are the measured distances from the point p_1 to the nodes n_1, n_2 and n_3.

$$(x_1 - s_1)^2 + (y_1 - t_1)^2 = R_1^2 \quad \textbf{(0)}$$

$$(x_1 - s_2)^2 + (y_1 - t_2)^2 = R_2^2 \quad \textbf{(0)}$$

$$(x_1 - s_3)^2 + (y_1 - t_3)^2 = R_3^2 \quad \textbf{(0)}$$

In this manner minimum of nine equations for the three points p_1, p_2, and p_3 can be obtained. [7] says that using Newton Equations of Motions another four equations revealing the relationship between the velocity (v) and acceleration (a), and distance between the pairs of points p_1-p_2 and p_2-p_3 can be obtained:

$$\sqrt{(x_1 - x_2)^2 - (y_1 - y_2)^2} = \tfrac{1}{2} \cdot a_1(\Delta t)^2 + v_1(\Delta t)^2 \quad \textbf{(0)}$$

$$\sqrt{(x_3 - x_2)^2 - (y_3 - y_2)^2} = \tfrac{1}{2} \cdot a_2(\Delta t)^2 + v_2(\Delta t)^2 \quad \textbf{(0)}$$

$$a_1 \cdot \Delta t = v_1 - v_0 \quad \textbf{(0)}$$

$$a_2 \cdot \Delta t = v_2 - v_1 \quad \textbf{(0)}$$

Additionally, it is also possible to obtain the following equations based on the compass output (z):

$$z_1 = \tan^{-1}\left(\frac{y_2 - y_1}{x_2 - x_1}\right) \quad \textbf{(0)}$$

$$z_2 = \tan^{-1}\left(\frac{y_3 - y_2}{x_3 - x_2}\right) \quad \textbf{(0)}$$

As stated in [6], [7], and [8] the key observation here is that there are more equations (15) than variables (12) that may have errors. Therefore, if one of the sensors is not functioning, the correct readings can be calculated from the system of equations.

2.4 Discussion

In this section, we give a brief summary of the techniques we have investigated for fault tolerance in wireless sensor networks. The two algorithms, M-FTA [1] and J/FTA [2] determine a reliable sensor reading from a set of faulty sensors. J/FTA technique is an improvement of M-FTA. In J/FTA, identifying possible faulty sensors for each interval concerns fault detection. On the other hand M-FTA would not detect any sensor as faulty in the single correct interval. [2] says that depending on the out put of J/FTA several inferences can be made about faulty sensors. Both techniques only address linear or 1-dimensional sensors. The authors say that the method presented can be extended to multi-dimensional sensors by replacing each interval corresponding to a physical value by a vector of intervals.

The two techniques [3] and [4] are able to detect unusual behavior about regions in an environment. [3] focuses on distinguishing between faulty sensor measurements and unusual environment conditions. The proposed

solution is based on a Bayesian fault-recognition algorithm and assumes that sensor faults are more likely to be uncorrelated and environment conditions are spatially correlated. On the other hand the techniques discussed in [4] focuses on the fault correction or fault tolerant detection problem in wireless sensor networks. [4] propose a distributed fault detection scheme as an optimization problem such that the optimal detection performance is achieved by optimizing the thresholds in sensors in the network and the fusion sensor. Additionally, [4] shows mathematically that the optimal fault detection error decreases exponentially with the increase of the neighborhood size.

Finally, we have introduce a technique proposed by [6], [7], and [8] to use one type of sensor to back-up sensors of different types. They have formulated the problem for two different types of sensors: binary and multi-level. An important feature of this approach is built-in-self-repair fault tolerance. [8] says it can be applied to a broad set of fault models, where a failure is an arbitrary type of inconsistence measurement by a sensor, which cannot be compensated systematically. In particular, it can be applied to sensor faults that cannot be corrected using calibration techniques.

The techniques we have discussed are developed to tolerate failures in different domains. For example, the solutions given in [3] and [4] are feasible solutions to tolerate failures in event detection but on the other hand it is impossible to apply these techniques to tolerate failures in sensors which track a trajectory of an object in a multi-dimensional space. In an event detection scheme, statistical methods are acceptable due to the distributed and correlated nature of the problem. The heterogeneous back-up schemes in [6],[7], and [8] can be employed to track objects or trajectory of objects in a multi-dimensional space. But it is not possible to apply a heterogeneous back-up scheme to tolerate sensor failures in an event detection scheme. The two techniques [1] and [2] tolerate only failures in 1-dimenstional linear sensors and hence cannot be assigned to a problem that deals with multi-dimensional sensors. Therefore, it is to see that the investigated techniques are domain oriented and it is difficult to find a common understanding and architecture among them.

3 Fault-Tolerance Techniques in Robotics

In this section we focus on fault-tolerance techniques in robotics. Fault tolerance is an important concern in the design and use of robotics [15]. In the past years, the military, nuclear power, and space programs have developed a number of techniques for fault tolerance in robots. In or discussion, first, we give a brief introduction to fault tree analysis (FTA), which analyze the structure of the robot and determines the flow of failures through the system [12]. Next, we describe the fault-tolerance capabilities and techniques in Hannibal robot [11] developed at MIT. Finally, we present a dynamic fault tolerant framework for robots developed at the Rice university [13].

3.1 Fault Tree Analysis

Fault Tree Analysis (FTA) is a deductive method in which failure paths are identified by using a fault tree drawing or graphical representation of the flow of fault events[13]. Each event in a tree is a component failure, an external disturbance, or a system operation[13]. The top event is usually the failure of the entire robot. A logical tree of failures is created by connecting events using logic symbols. Some of the basic symbols are explained in the Table 1 below:

Symbol		Function
	AND gate	All inputs required to produce output event
	OR gate	Any one input event causes the output event
	Rectangle	A malfunction which results from a combination of fault events through logic gates
	Diamond	A fault event for which the causes are left undeveloped
	Circle	A basic fault event, whose frequency and failure mode are known
	Triangle	A suppressed tree. The tree is detailed in another figure

Table 1. Fault Tree Analysis Symbols

A fault tree is developed in a top down manner [12]. The top event is broken into primary events, which cause the failure at the top through some logical combinations. This process is repeated to deeper levels until a basic event or an undeveloped event is reached[12].

It is stated in [12] that the information available in the fault trees may be enhanced by a quantitative analysis of the failures. Here, each input event is assigned with a failure rate and based on the rules of the connecting logic gates the failure rates are propagated up the tree. For example, the output of an OR gate is the some of the inputs and the output of an AND gate in the product of its inputs. The failure probabilities allow the fault tolerance design of a robot to focus on the components which are more likely to fail.

As shown in Fig. 10 , we present the fault tree of a robot from [13]. Here, we only present the fault tree for the overall robot failure. The top event is the failure of the entire robot. Power failure, computer failure, or a combination of failures of the joints, are the primary causes of the robot failure.

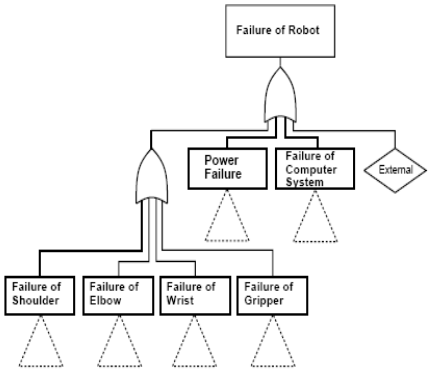


Fig. 10. Robot fault tree

These primary failures are further represented using fault trees making a top down representation of the propagation of the fault events. If the robots joints are redundant, it can withstand the failures of the joints and still continue its tasks. [12] says, if the robot has two redundant degrees of freedom, it can tolerate as many as two joint failures.

3.2 Failure Recognition and Fault Tolerance Techniques in Hannibal Robot

Hannibal is a small autonomous robot developed in the Artificial Intelligence lab at MIT. It has 19 actuators and over 60 sensors of 5 different types all connected via a local network to 8 on-board computers [11]. Most of the failures in Hannibal are caused by sensor and actuator faults. These faults effect various levels of Hannibal's system hierarchy[11]: hardware level, low level control where sensor information is processed, and high level control which determines the robot's behavior. The sensor failures directly affects the hardware level, they in turn affects the virtual sensors in the robots low level control, and finally affecting high level control making unusual or faulty behavior of the robot.

Fault tolerance is implemented on Hannibal using a distributed network of concurrently running processes[11]. Therefore, for each component a set of fault tolerance processes are written to tolerate hardware failures. [11] says that it is important to detect and confine errors to the lowest possible level in which they occur. Next, we briefly describe possible fault-tolerance techniques at each level and the implementation of fault-tolerance in Hannibal.

3.2.1Fault Tolerance at Different Levels

[10] suggests replication of hardware, redundant control behaviors, and robust virtual sensors as possible fault-tolerance strategies for each level. Here, we look at these strategies for each level in more detail.

3.2.1.1Replication of Hardware

Here, replication of hardware corresponds to replicating sensors and actuators. As stated in [10], for example, multiple potentiometers could be used to sense a particular joint angle and an arbiter could gather the joint angle values of each potentiometer and accept the most common joint angle value as correct, see Fig. 11 .

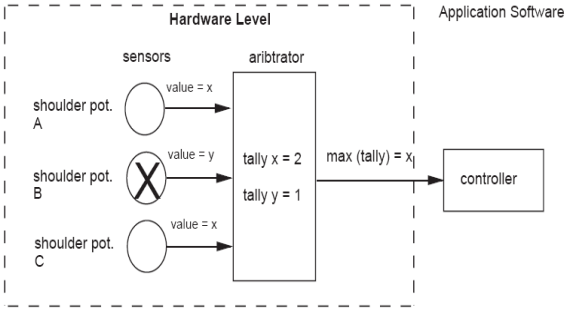


Fig. 11. Replication of sensors in hardware level

[10] says this approach is impractical to enhance fault-tolerance in Hannibal. Because of Hannibal's size and weight constraints restrict mounting on sensors on it.

3.2.1.2Redundant Control Behaviors

This strategy implements fault-tolerance in high level control and addresses high level failures. Here, the controller is designed with redundant strategies to perform a task[10]. If a failure occurs in one strategy, then the controller eventually tries another strategy. But [10] says that this approach is not well suited for hardware failures. The robot must function in hazardous environments and remaining the robot unstable till the controller finds a stable strategy is not acceptable.

3.2.1.3Robust Virtual Sensors

Robust virtual sensors are virtual sensors which remain reliable despite sensor failures[10]. This approach is suited for local failures, which affects only one part of the robot, for example, a failure in a robots leg. On the other hand, this approach is not suitable for some failures that affect the overall behavior of the system, so called global failures.

3.2.2Fault-Tolerance in Hannibal

Hannibal's fault tolerance capabilities are implemented using adaptive agents [11]. For each sensor an adaptive virtual sensor is implemented. The adaptive sensors maintain reliable performance upon a failure of a sensor. For example, the appropriate virtual sensors are informed, when a failure is detected. The virtual sensors then respond by reconfiguring the way they use their sensory information. [11] says that this approach in Hannibal also exploits adaptation instead of redundancy. For example, if a robots leg is broken, the high level control is informed to change the control that the motion of the robot remains stable with one less leg. Whereas a redundant approach might involve trying different walking strategies.

3.3 Dynamic Fault-Tolerance Framework for Robots

In this section, our discussion is based on a multi-layer intelligent control framework for fault-tolerance in remote robots. [14] says that fault tolerant control of a robot system can be divided into three layers: a continues servo layer, an interface monitor layer, and a discrete supervisor layer as shown in Fig. 12. The servo layer consists of the normal robot controller and the robot. The interface layer provides fault detection and some basic

fault tolerance for the system. The main task of the supervisor layer is to provide higher level fault tolerance and general action commands for the robot. Next, we describe the functionality of each layer in the framework.

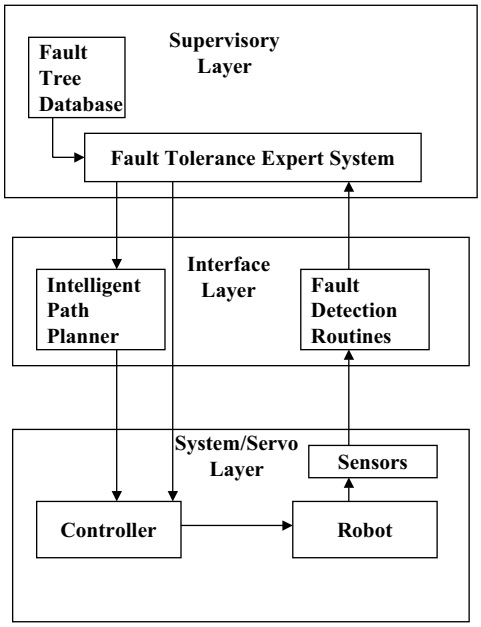


Fig. 12. Multi-layer fault tolerant robot system

3.3.1Continues Servo Layer

This layer contains robot and the robot control computer [14]. Typically most of the robots are modeled using the dynamic equations given in [14]. The robot controller uses the sensed estimates of joint positions to compute the necessary torque to apply to each motor in order to move the robot from one position to another. The controller uses internal sensors to obtain the information about the current position or velocity of the robots joints. It also uses sensors to determine the errors between desired and sensed values of joint positions. Therefore, if a sensor is damaged and the failure remains undetected, the controller will receive incorrect information about the joint and will result in an unusual behavior of the robot[12].

3.3.2Interface Monitor and Fault Detection Layer

The interface layer contains the robot planner and fault detection routines. Fault detection routines are based on the concept of analytical redundancy described in [9]. The fault detection routines consists of two major stages: detection tests and decisions. The basis of the detection tests is analytical redundancy, which takes two forms: direct redundancy- the relationship among instantaneous outputs of sensors and temporal redundancy- the relationships among the histories of sensor outputs and actuator inputs[9]. Based on these relationships the outputs of dissimilar sensors at different times can be compared. These comparisons are then measures of the dissimilarity between the behavior of observed sensor outputs and the behavior that should be result under normal conditions[9]. For example, as stated in [9], if we consider the relationship between velocity (v) and acceleration (a):

$$v(k+1) = v(k) + Ta(k) \qquad (\mathbf{0})$$

Where T is the sampling interval. Equation 18 provides a way of comparing velocity measurements and accelerometer outputs and can be used for the detection of both types of sensor failures. The detection routine performs similar tests to detect sensor errors and finally detects joint failures in the robot.

3.3.3Supervisor and Fault Tolerance Layer

In this layer a fault-tolerance expert system performs robotic fault tolerance using fault trees[14]. As we have already discusses, a fault tree is a standardized representation of the robots fault structure. The failed components that are reported from the detection routines in the interface layer are pruned by the expert system[14]. The pruning is also supported by time varying failure probabilities.

3.4 Discussion

The fault tree analysis [12] is a deductive method in which failure paths are identified in a robot. [12] says that it has proven useful in pointing out the interaction between failures in a robot system. Fault trees are also useful to design fault tolerance capabilities of robots. It is stated in [12] that through an analysis of the robot structure displayed by the trees, dynamic reconfiguration strategies can be developed which maintain the robot in the presence of failures.

In [10], the fault tolerance capabilities of the robot are implemented using adaptive agents. This strategy concentrates on adaptation rather than redundancy. To implement this strategy faults in various levels in a robot are analyzed to detect and tolerate them at the lowest possible level in which they occur. Here, the robot dynamically tailors its use of sensors and actuators to minimize the failures on it performance. Due to adaptation the robot is capable to dynamically reconfigure itself with the available components.

The dynamic fault tolerance framework introduced in [13] addresses the problem within a layered architecture. Here, once a failure is detected in a lower layer, a fault tolerant expert system combining a fault tree in a higher level take appropriate actions to make use of the existing robot structure, redundancy, and alternate paths.

We identify that fault tree analysis as a useful technique to identify and classify failures in any robot system. The framework introduced in [13] uses a fault tree connected to an expert system to implement fault tolerance capabilities. The adaptive algorithm introduced in [10] is a robust technique to implement fault tolerance in small robots that should operate in a hazardous environment. Given that the robot is small in size, it constraints replicating sensors in the robot. Therefore, the adaptive strategy is an acceptable solution. Even one should consider the constraints for applying a layered framework as in [13] for a small robot. In [13], fault detection is based on analytical redundancy, which requires various type of replicated sensors. Therefore, when applying such a framework for a small robot, the fault detection routines should be based on limited constrained resources. On the other hand the dynamic framework in [13] is a systematic solution to fault tolerance in a robot system. The layered architecture allows to use different fault detection and tolerance techniques at different levels.

4 Classification of the Fault-Tolerance Techniques

A classification of the investigated fault-tolerance techniques is shown in Table 2. First, we have classified the techniques analyzing the level in the system in which fault-tolerance takes place. Here, we have identified that the techniques we have investigated are either embedded in the hardware level or in the application level. We have seen in the hardware level as well as in the application level, some techniques only detect faults and some only tolerate faults without considering any detection. We have also seen that some techniques are based on both fault-detection and tolerance. Therefore, next, we classify the techniques based on their fault-detection or fault-tolerance capabilities. Finally, we classify further the techniques in the application level according to the type of the solution: analytical or statistical.

The two algorithms introduced in [1] and [2] determine a reliable sensor reading from a set of faulty sensors. We classify both as application level techniques. [1] does not identify faulty sensors but on the other hand [2] is an improvement of [1], which identifies possible faulty sensors. Since techniques in [1] and [2] are based on graphs we classify them as analytical solutions.

The two techniques in [3] and [4] for fault-tolerance in event region detection are based on statistical models in the application level. The solution in [3] is based on a Bayesian fault-recognition algorithm and focuses only on fault-detection. On the other hand the techniques discussed in [4] focus on fault tolerant detection problem in wireless sensor networks.

The heterogeneous backup schemes discussed in [6], [7] and [8] for binary and multi-level sensors detect faults in the hardware level and propose a fault-tolerance scheme in the application level. Since these solutions are based on optimizing techniques, we can classify them as analytical solutions.

Now, we classify the fault-tolerance techniques we have introduced for robotics. In [12], we investigate fault tree analysis, which analyze the structure of the robot and determines the flow of failures through the system. Fault trees support fault-tolerant design and development of robots. We identify it as an analytical as well as a statistical technique, since information available in the fault trees may be enhanced by statistical analysis of failures. Fault tree analysis operates on a robots high level control level, therefore we classify it as an application level solution.

The fault-tolerance capabilities implemented in [10] for a small autonomous robot are based on adaptive agents. It detects faults in the hardware level and the application level responds to the failures by reconfiguration. Therefore, in [10], the fault-tolerance capabilities are implemented analytically at application level and some level of fault-tolerance reconfiguration is also done at hardware level.

The final technique we investigate, [13] is a fault-tolerance framework for a robot. Here, fault detection is performed at hardware level using analytical redundancy. In [13], fault tolerance capabilities are implemented in the high level control of the robot (application level) using fault trees.

Technique	Hardware Level		Application Level			
	Fault-Detection	Fault-Tolerance	Fault-Detection		Fault-Tolerance	
			Analytical	Statistical	Analytical	Statistical
[1]					X	
[2]			X		X	
[3]				X		
[4]				X		X
[6,7,8]-binary	X				X	
[6,7,8]-multilevel	X				X	
[10]	X	X			X	X
[12]			X	X		
[13]	X				X	

Table 2. Classification of the investigated techniques

5 Conclusions

In our survey, we have introduced some fault tolerance and detection techniques that are employed in wireless sensor networks and robotics. Replication of hardware, sensors in sensor networks and sensors and actuators in robotics can be identified as the ground solution to the problem. Nevertheless, based on redundancy, different approaches to tolerate failures in both fields have been proposed.

As compared to sensor networks, fault tolerance techniques in robotics address the problem in a more systematic way by layering fault detection and tolerance capabilities at different levels in the system. It is also enriched by techniques like fault tree analysis. Adapting a layered fault tolerance architecture, where different fault tolerance protocols are running on different levels in a sensor network would be a major research direction in the future. This type of an architecture could enhance a heterogeneous back up scheme that is running on a higher level in the sensor network, where lower levels in the system perform fault detection based on statistical or analytical processing. With the future development of hardware, for example, power efficient sensors with high processing capacity, such layered fault tolerance frameworks for wireless sensor networks would be possible to be realized.

6 References

[1] Keith Marzullo, Tolerating failures of continuous-valued sensors, ACM Transactions on Computer Systems (TOCS), v.8 n.4, p.284-304, Nov. 1990

[2] Jayasimha, D.N., "Fault tolerance in multisensor networks," *Reliability, IEEE Transactions on* , vol.45, no.2pp.308-315, 320, Jun 1996

[3] Krishnamachari, B. and Iyengar, S. 2004. Distributed Bayesian Algorithms for Fault-Tolerant Event Region Detection in Wireless Sensor Networks. *IEEE Trans. Comput.* 53, 3 (Mar. 2004), 241-250

[4] Huang, Y. 2006. On Distributed Fault-Tolerant Detection in Wireless Sensor Networks. *IEEE Trans. Comput.* 55, 1 (Jan. 2006), 58-70

[5] Q. Zhang, P.K. Varshney and R.D. Wesel, [ldquo]Optimal Bi-Level Quantization of i.i.d. Sensor Observations for Binary Hyperthesis Testing,[rdquo] IEEE Trans. Information Theory, vol. 48, no. 7, pp. 2105-2111, 2002

[6] Koushanfar, F.; Potkonjak, M.; Sangiovanni-Vincentell, A., "Fault tolerance techniques for wireless ad hoc sensor networks," *Sensors, 2002. Proceedings of IEEE* , vol.2, no.pp. 1491- 1496 vol.2, 2002

[7] F. Koushanfar, S. Slijepcevic, M. Potkonjak, A. Sangiovanni-Vincentelli, “Error-Tolerant Multimodal Sensor Fusion.” IEEE CAS Workshop on Wireless Communications and Networking, September 2002

[8] F. Koushanfar, M. Potkonjak, A. Sangiovanni-Vincentelli. “Fault-Tolerance in Sensor Networks.” Book chapter, in: ‘Handbook of Sensor Networks’, I. Mahgoub and M. Ilyas (eds.), CRC press, Section VIII, no. 36, 2004

[9] Chow, E.; Willsky, A., "Analytical redundancy and the design of robust failure detection systems," *Automatic Control, IEEE Transactions on* , vol.29, no.7pp. 603- 614, Jul 1984

[10] Ferrell, C. 1993 Robust Agent Control of an Autonomous Robot with Many Sensors and Actuators. Technical Report. UMI Order Number: AITR-1443., Massachusetts Institute of Technology

[11] Ferrell, C. 1994. Failure recognition and fault tolerance of an autonomous robot. Adapt. Behav. 2, 4 (Mar. 1994), 375-398

[12] Visinsky, M. L., Cavallaro, J. R., and Walker, I. D. (1994). Robotic fault detection and fault tolerance: a survey. Reliability Eng. and System Safety, 46:139–158

[13] Visinsky,M. L. (1991). Fault detection and fault tolerance methods for robotics. Master’s thesis, Rice University.

[14] Visinsky, M.L.; Cavallaro, J.R.; Walker, I.D., "A dynamic fault tolerance framework for remote robots," Robotics and Automation, IEEE Transactions on , vol.11, no.4pp.477-490, Aug 1995

[15] J. R. Cavallaro and I. D. Walker. A Survey of NASA and Military Standards on Fault Tolerance and Reliability Applied to Robotics. In Proc. AIAA/NASA Conference on Intelligent Robots in Field, Factory, Service, and Space, pages 282-286, Houston, TX, March 1994

[16] Avizienis, A.; Kelly, J.P.J., "Fault Tolerance by Design Diversity: Concepts and Experiments," Computer , vol.17, no.8pp. 67- 80, Aug. 1984

[17] Tanenbaum, A. S. and Steen, M. V. 2001 Distributed Systems: Principles and Paradigms. 1st. Prentice Hall PTR.

[18] Harte, S.; Rahman, A.; Razeeb, K.M., "Fault tolerance in sensor networks using self-diagnosing sensor nodes," Intelligent Environments, 2005. The IEE International Workshop on (Ref. No. 2005/11059) , vol., no.pp. 7- 12, 29 June 2005

Energiesparmechanismen für Ubicom-Plattformen

Steffen Melischko

Telecooperation Office, Institut für Telematik, Universität Karlsruhe (TH)
steffen_melischko@web.de

Abstract: Der Einsatz energiesparender Mechanismen in batteriebetriebenen Systemen, wie beispielsweise in Sensornetzwerken, dient zum einen der Verlängerung der Einsatzfähigkeit dieser Systeme, sowie der effizienten Nutzung der zur Verfügung stehenden Energie. In dieser Seminararbeit werden verschiedene Konzepte für ein energiesparendes Design vorgestellt und bewertet.

1 Einführung

Energiesparmechanismen in batteriebetriebenen Ubicomp-Plattformen haben die Aufgabe, die begrenzt zur Verfügung stehende Energie optimal auszunutzen, um so die Lebenszeit bzw. Verfügbarkeit dieser Systeme zu verlängern. Gerade im Bereich von Sensornetzwerken ist die Maximierung der Lebenszeit von größter Bedeutung, da meist eine große Anzahl von Sensoren verwendet wird, die zudem noch oft in schwer zugänglichen Umgebungen platziert werden und somit den Austausch von Batterien zu teuer oder unmöglich macht. Die zur Verfügung stehenden Mechanismen lassen sich grob in drei Kategorien unterteilen:

1. Batterie-zentrische Ansätze
   Die Batterie und ihre charakteristischen Eigenschaften wie beispielsweise das Verhalten bei hochohmiger Entladung oder dem Ausnutzen von Recovery-Effekten stehen hier im Mittelpunkt.
2. Scheduling-zentrische Ansätze
   Modellierung des Energieverbrauchs aus der Sicht der Systemsoftware, wie beispielsweise der Energiebedarf einer Task oder einer ganzer Anwendung.
3. Duty-cycle Ansätze
   Ziel bei diesen Ansätzen ist das Einfügen von Ruhepausen nach einer Arbeitsphase. Ein System soll nur dann in einem aktiven Zustand sein, wenn es auch tatsächlich Arbeit zu erledigen hat.

Diese Ansatzmöglichkeiten werden im Folgenden beschrieben und bewertet, sowie explizite Lösungsansätze für die einzelnen Kategorien vorgestellt. Zu Beginn meiner Ausarbeitung werde ich auf die Funktion einer Batterie sowie ihrer Charakteristischen Merkmale eingehen, danach analysiere ich die verschiedenen Klassen von Energiesparmechanismen und werde die Vor- und Nachteile der verschiedenen Ansätze herausarbeiten. Am Ende werde ich anhand der Beispielplattform COBIS (Collaborative Business Items) den praktischen Einsatz von Energiesparmechanismen aufzeigen.

2 Batterien und ihre charakteristische Eigenschaften

2.1 Grundlegender Aufbau und Funktion einer Batterie

Eine Batterie ist tragbare chemische Energie, die in elektrische Energie umgewandelt wird, sobald sie in einen Stromkreis eingesetzt und entladen wird [1].
Hauptbestandteile einer Batterie sind jeweils eine positive und negative Elektrode, eine Elektrolytlösung, sowie ein Seperator. Der grundlegende Aufbau einer Batterie wird in Abb. 1 anhand einer Bleibatterie verdeutlicht. Beide Elektroden bestehen aus massiven Bleistäben, die als Stromsammler bzw. Ableiter dienen. Auf die Bleistäbe wird jeweils poröses aktives Material aufgetragen (pos. Elektrode = Bleidioxid, neg. Elektrode = Pb-Schwamm), welches den eigentlichen chemischen Energiespeicher darstellt. Beide Elektroden sind vollständig in die Elektrolytlösung eingetaucht und durch einen Seperator voneinander getrennt, um einen Kurzschluss beider Elektroden zu verhindern. Hauptsächlich dient die Elektrolytlösung als Ionenleiter, aber im Falle der Bleibatterie auch als Reaktionspartner bei der Entlade- und Ladereaktion. Sobald nun die Batterie in einen Stromkreis eingesetzt wird, werden Plus- und Minuspol über einen äußeren Kontakt miteinander verbunden und der Prozess der Elektrolyse beginnt. Hier reagieren Metall und Metalloxid miteinander, wobei sich die chemische Energie freisetzt. Diese Reaktion nennt man auch Redoxreaktion (Metall wird oxidiert und Metalloxid reduziert). Bei

diesen Reaktionen wird elektrische Ladung von einem Stoff auf den anderen übertragen. Der elektrische Strom besteht aus fließenden Elektronen, die sich infolge dieser elektrischen Spannung als treibende Kraft bewegen. Einfach ausgedrückt bietet die negative Elektrode die Elektronen in großer Stückzahl und mit hohem Druck an, und die pos. Elektrode saugt die Elektroden ab. Der Druckunterschied in der Batterie entspricht der Spannung und die fließende Menge an Elektronen pro Zeiteinheit ist der Strom.

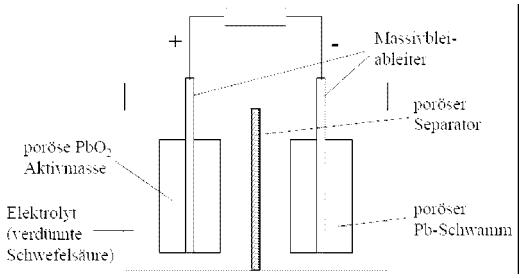


Abb. 1 schematischer Aufbau einer Batterie [2]

2.2 Batteriemodelle

S. Park et al. nahmen in [3] eine Kategorisierung der Batteriemodelle dahingehend vor, indem sie zwischen einem Linearen-, einem Discharge-Rate-Dependent- und einem Relaxation-Modell unterschieden. Diese Unterteilung berücksichtigt zum einen das Verhalten einer Batterie bei hochohmiger Entladung und zum anderen auch die Erholung während Ruhepausen. Bei allen vorgestellten Modellen geht es darum, die aktuelle Restkapazität der Batterie so genau wie möglich zu berechnen.

Die maximale Kapazität wird in Ampere * Hour [Ah] angegeben. Diese Angabe ist aber nur theoretischer Natur, da die reale maximale Kapazität von sehr vielen und komplexen Batterieeigenschaften abhängt.
Mit Hilfe des momentanen Entladestroms I und der maximalen Kapazität C kann die Batterielebenszeit T nach (1) berechnen.

$$\text{Battery-Lifetime} = C / I \qquad (\mathbf{1})$$

2.2.1 Lineares Modell

Mit der aktuellen Kapazität C' der Batterie, der Entladezeit t_d mit Stromstärke I(t) lässt sich nun die Restkapazität C nach (2) berechnen.

$$\text{Restkapazität } C\,[Ah] = C' - \int_{t=t_0}^{t_0+t_d} I(t)dt = C' - I * t\Big|_{t_0}^{t_0+t_d} = C' - I * t_d \qquad (\mathbf{2})$$

In diesem Modell wird die Batterie als linearer Stromspeicher angesehen, d.h. man geht von einer konstanten Stromstärke I(t) während der Zeit t_d aus. Dieser Ansatz ist sehr einfach und bietet die Möglichkeit den Energieverbrauch einer Anwendung zu messen, jedoch ohne das reale Verhalten einer Batterie mit ein zu beziehen. So nimmt dieses Modell keine Rücksicht auf die Höhe der Entladung in den jeweiligen Intervallen, die eine wichtige Rolle in der Annäherung an die tatsächliche Vorhandene Kapazität spielt. Des Weiteren wird auch der Recovery-Effekt (siehe 2.2.3) einer Batterie nicht berücksichtigt.

2.2.2 Discharge Rate Dependent Model

In [4] [5] [6] wurde gezeigt, dass höhere Entladeströme die tatsächlich nutzbare Kapazität vermindern. Je höher die Belastung, desto niedriger ist die zur Verfügung stehende Kapazität. Dieser Effekt wird auch als Rate-Capacity-Effekt bezeichnet. Der Grund für diesen Effekt ist im inneren der Batterie zu suchen. Sollte die Belastung über dem vom Hersteller angegebenen Wert liegen, so reagieren nur noch die äußeren Bereiche an der Kathode mit den positiven geladen Teilchen der Anode. Im inneren der Kathode sind aber noch Bereiche verfügbar, die ebenfalls für die Reaktion genutzt werden könnten. Diese sind nun aber nicht mehr erreichbar. Somit steht theoretisch noch Kapazität zur Verfügung, die aber praktisch nicht genutzt werden kann. Um dieser Erkenntnis gerecht zu werden, wird ein Faktor k eingeführt (3), der die Effizienz der Batteriekapazität darstellt.

$k = C_{eff} / C_{max}$; C_{eff} = effektive Batteriekapazität **(3)**

Somit ergibt aus (2) und unter der Berücksichtigung von k die Gleichung (4)

$C\ [Ah] = k*C'-I*t_d$ **(4)**

Der neu hinzugekommene Faktor k variiert mit der Stromstärke I und berücksichtigt somit die Tatsache, dass sich bei hochohmiger Entladung die Restkapazität schneller verringert, als beim linearen Ansatz. Eine Möglichkeit, um den Faktor k zu bestimmen, sind die Datenblätter der jeweiligen Hersteller. Diese Daten werden dann graphisch wie in Abb.1 dargestellt, indem man die effektive Kapazität zu den verschiedenen Entladeströmen abbildet. Allerdings berücksichtigt dieses Modell noch nicht den Recovery-Effekt.

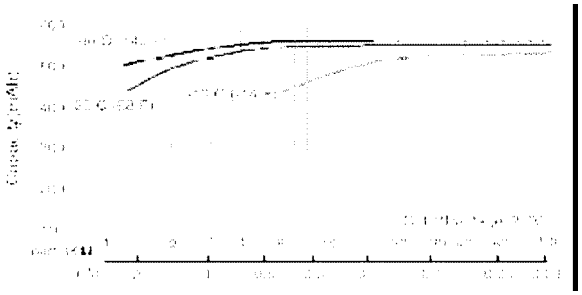


Abb. 2 Capacity vs. Discharge Rate Curve for CR 2354 [3]

2.2.3 Relaxation Model

Wenn Batterien mit großen Strömen entladen werden, nimmt die Diffusionsrate der aktiven Materialen stark ab. Wird diese hohe Entladung über längere Zeit beibehalten, dann kann die Batterie die erforderliche Energie nicht mehr liefern, obwohl noch genügend aktives Material vorhanden ist. Wird die Entladungsrate jedoch gesenkt oder keine Energie mehr benötigt, dann nimmt die Diffusions- und Transportrate des aktiven Materials wieder zu und die Batterie holt sich die Kapazität wieder zurück , die sie durch die hohe Entladung zusätzlich verloren hat. Dieser Effekt wird „Recovery-Effekt“. genannt. Der Grund für den Recovery-Effekt ist, dass sich die Ionen im Elektrolyt gleichmäßig verteilen, wenn keine Belastung vorliegt und somit eine möglicherweise entstandene Ungleichverteilung während der Entladung wieder neutralisiert wird. In [4] [5] [7] wurde dieses Phänomen analytisch und stochastisch untersucht. Abbildung 3 stellt die Auswirkung des Recovery-Effektes graphisch dar.

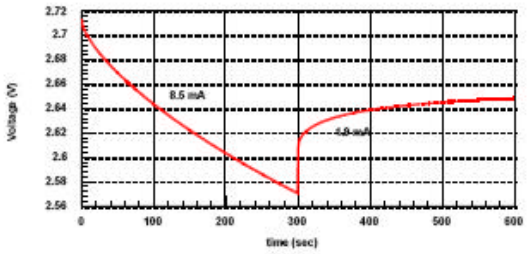


Abb. 3 Recovery-Effekt of Relaxation Model [3]

3 Energiesparmechanismen

Jedes batteriebetriebene Gerät wie beispielsweise Handys, PDA's kann nur so lange seine Dienste anbieten und Aufgaben erledigen, so lange seine Batterie in der Lage ist, die notwendige Energie dafür zu liefern. Somit besteht ein immer größer werdendes Interesse daran, Mechanismen für eine effizientere Nutzung der Batterie einzusetzen, um letztendlich die Lebenszeit der Geräte zu verlängern. Effiziente Energiesparmechanismen setzten aber exakte Batteriemodelle voraus (siehe 2.2), um den Energiebedarf eines Gerätes zu einem bestimmten Zeitpunkt zu berechnen und somit Aussagen über die verbleibende Restenergie in der Batterie machen zu können. Ziel aller hier vorgestellten Mechanismen ist es, die Differenz zwischen der theoretischen Lebenszeit (vom Hersteller angegeben) und der realen Lebenszeit (Dauer bis Batterie unter realem Einsatz nicht mehr in der Lage ist, die geforderte Energie zu liefern) zu minimieren.

3.1 Batterie-zentrische Ansätze

Im Mittelpunkt der Batterie-zentrischen Ansätze steht die Ausnutzung des „Recovery-Effektes". Letztendlich versuchen alle hier vorgestellten Ansätze, der Batterie Zeiten der Erholung zur Verfügung zu stellen. Zum einen kann man diese Erholungszeiten damit erreichen, indem man von einem Ein-Batterie-System zu einem Mehr-Batteriesystem über geht und somit die geforderte Energie auf mehrere Batterien verteilt oder die Batterien im Wechsel an den Stromkreis aufschaltet. Im Falle eines Ein-Batterie-Systems nutzt man hauptsächlich den Mechanismus der gepulsten Entladung (3.1.2.1). Sollten jedoch die Anforderungen des Systems das Einfügen von Ruhepausen im Sinne der gepulsten Entladung nicht ermöglichen, wird unter anderem der Ansatz der Frequenzskalierung verwendet, um die Batteriekapazität im Sinne der Lebenszeitmaximierung effizient auszunutzen.

3.1.1 Mehr-Batterie-Systeme

Im Bezug auf Mehr-Batterie-Systeme haben frühere Untersuchungen gezeigt, dass eine sequentielle Entladung im Sinne der Verlängerung der Batterielebenszeit, keine befriedigenden Ergebnisse liefert. So wurde in [8] eine Variante der sequentiellen Entladung für ein Zwei-Batteriesystem vorgestellt, die immer die Batterie auswählt, die am Besten mit der geforderten Stromlast umgehen kann. Diese Strategie ist aber nur dann effektiv, wenn man asymetrische Batterien verwendet (eine Batterie arbeitet sehr effizient bei niedrigem Energiebedarf, die andere bei großem Energiebedarf) und wenn das System auf unterschiedlichen Energieniveaus arbeitet. Verwendet man hingegen baugleiche Batterien oder fordert das System nur ein konstantes Energieniveau, so degeneriert dieses Schedule zu einer rein sequentiellen Entladung.

3.1.1.1 Fast Alternation between Heterogeneous Cells

L. Benini et al. verfolgt in [9] den Ansatz, die geforderte Stromlast auf mehrere Batterien zu verteilen, um sie so gleichzeitig aber nur teilweise zu entladen. Die gleichzeitige Entladung wird durch eine hochfrequente Alternierung zwischen den zur Verfügung stehenden Batterien erreicht, wie sie auch schon in [10] dargestellt wurde. Ziel ist es bei [10] mittels sehr hohen Frequenzen zwischen zwei baugleichen Batterien hin und her zu schalten, um so eine Batterie mit doppelter Kapazität zu simulieren. Dies wirkt dem Nachteil entgegen, eine Batterie mit großen Strömen zu entladen. Jedoch werden in [9] heterogene Batterien verwendet, um auf unterschiedliche nominelle Kapazitäten und unterschiedliches Entladeverhalten der Batterien zurückgreifen zu können, um effizient mit unterschiedlichen Entladeströmen umgehen zu können. Da nun heterogene Batterien verwendet werden, wird die geforderte Energie nicht auf alle Batterien gleich verteilt, sondern die Batterien werden nach einer Round-Robin-Politik für Zeitperioden unterschiedlicher Dauer aufgeschaltet. Es werden keine festen Perioden verwendet, da die zu entnehmende Energie von der aktuellen Kapazität der einzelnen Batterie abhängt. Um nun den richtigen Entladestrom für jede Batterie zu ermitteln, wird das Problem des passenden Entladestroms in ein kontinuierliches abhängiges Optimierungsproblem umtransformiert, welches man dann mit Standartoptimierungsmethoden lösen kann. Als Vorraussetzung für die Lösung dieses Problems muss jedoch bekannt sein, welche unterschiedlichen Stromstärken das System benötigt und deren prozentuale Verteilung während der Systemlaufzeit. Um dieses Optimierungsproblem zu lösen wurde hier das „Sequential Optimization Packet" von MATLAB verwendet. Es wurden Experimente mit künstlich erzeugten Workloads, sowie ein Experiment an einem digitalen Audiorecorder durchgeführt. Bei jedem der Versuche konnte man sehen, das man die Lebenszeit der Batterien um bis zu 12% im Vergleich zu anderen Batterie-zentrischen Ansätzen verlängern kann und im Vergleich zur rein sequentiellen Entladung um bis zu 160%. Jedoch muss man diese Ergebnisse mit Vorsicht betrachten, da man hier im Voraus die prozentuale Verteilung der geforderten Systemstromstärken kennen muss, was nur bei Systemen mit statischen Workloads der Fall sein dürfte. Des Weiteren macht man sich bei diesem Ansatz den Recovery-Effekt nicht zu nutzen, da dieser auf Grund der hohen Frequenz beim Wechsel der Batterien so gut wie keine Verbesserung herbeiführen kann. Am Rande sei noch erwähnt, dass keinerlei Angaben darüber gemacht werden, wie viel Energie der hochfrequente Wechsel der Batterien benötigt.

3.1.1.2 Terminal Voltage Based Battery Scheduling
Einen dynamischen energiesparenden Ansatz für Multi-Batterie-Systeme stellen K. Lahiri in [11] vor. Hier wird das „Terminal Voltage Based Battery Scheduling“ vorgestellt, welches eine Round-Robin-Strategie verwendet, die nur die Batterien berücksichtigt, deren Ausgangsspannung über einem gegebenen Grenzwert liegt. Fällt die Spannung einer Batterie unter den Grenzwert, so wird sie aus der Round-Robin-Strategie heraus genommen. Wenn sich die Batterie soweit erholt hat, dass sich die Spannung wieder über dem Grenzwert befindet, wird sie wieder in die Strategie mit einbezogen. Allerdings benötigt man nun zusätzliche Systemberechnungen, die die Ausgangsspannung der Batterien misst und dann jedes Mal ein neues Set von verfügbaren Batterien berechnet. Diese Berechnungen müssen auch in den Energiebedarf des Systems mit einberechnet werden. Im Vergleich zum obigen statischen Round-Robin-Strategie bietet dieser Ansatz aber die Möglichkeit, eine Batterie mit niedriger Kapazität eine lange Erholungspause zu geben, indem man sie aus dem Set der aktiven Batterien heraus nimmt.

3.1.2 Ein-Batterie-Systeme
Während man bei den Mehr-Batterie-Systemen die benötigte Energie auf mehrere Batterien verteilen konnte, werden bei Ein-Batterie-Systemen Mechanismen wie die gepulste Entladung oder die Frequenzskalierung verwendet.

3.1.2.1 Pulsed Battery Discharge
C. F. Chiasserini und R. R. Rao nähern sich dem Problem der Lebenszeitmaximierung von Ein-Batterie-Systemen in [7] über die gepulste Entladung von Batterien. Der Prozess von der voll aufgeladenen zur entladenen Batterie wird hier in einem stochastischen Modell mittels Bernoulliprzessen modelliert. Grundprinzip ist die Belastung der Batterie für einen kurzen Zeitraum, gefolgt von einer Ruhepause, um so auch wieder den „Recovery-Effekt“ auszunutzen. Die Entladung erfolgt in stochastischen Momenten (Zustandsübergang), sowie auch die Erholung, wenn keine Energie benötigt wird. Getestet wird die Paketversendung in Kommunikationsgeräten einmal mit konstanter Stromentladung und das andere Mal mit gepulster Entladung, um die Ergebnisse vergleichen zu können. Annahmen wurden dahingehend gemacht, dass man von einer Batterie mit einer Zelle ausgeht und eine Zeiteinheit einem Frame entspricht. Des Weiteren wird nur der Energieverbrauch betrachtet, der zum Versenden eines Paketes notwendig ist (charge unit), so genannte Hintergrundströme wie beispielsweise der Stromverbrauch, wenn kein Paket versendet wird, werden vernachlässigt. Für die bessere Bewertung der gepulsten Entladung werden einmal die binäre (in jedem Zeitframe wird ein Packet versendet, wenn eins da ist, wenn nicht kann sich die Batterie um eine Einheit erholen, allerdings wird beim Versenden während des gesamten Zeitframes Strom benötigt) und zum anderen die allgemeine gepulste Entladung (in einer Zeiteinheit kann man ein oder mehr Pakete versenden oder/und eine Einheit Energie zurück gewinnen (Recovery Effekt)) verwendet. Es kann aber nur so langer Energie zurück gewonnen werden, bis die theoretische Kapazität verbraucht ist.

- Binär gepulste Entladung
  Der Prozess der Entladung beginnt bei N (Batterie voll aufgeladen) und endet bei 0, außer die theoretische Kapazität ist vorher schon erschöpft. N_s stellt einen „dummy-state“ da, der einfach den Beginn der Entladung darstellt. Der Zustand N kann mehrmals durch Aufladeprozesse erreicht werden.

  $a_1 = q$ entspricht der Wahrscheinlichkeit, dass ein Packet in einem Zeitrahmen gesendet wird
  $a_0 = (1-q)$ entspricht der Wahrscheinlichkeit für Ausnutzung des Recovery-Effektes

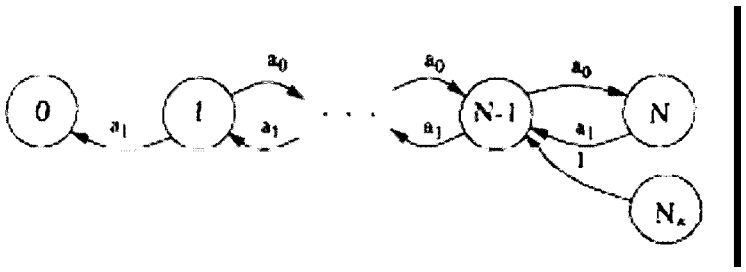


Abb. 4 Binäre Markovkette, die das Zellverhalten widerspiegelt [7]

- Allgemeine gepulste Entladung
  In dieser Variante sind Bursts von Paketen erlaubt. a_k ist die WS das ein ankommender Burst k Pakete enthält. M ist die Maximalanzahl von Paketen, die pro Frame verschickt werden.

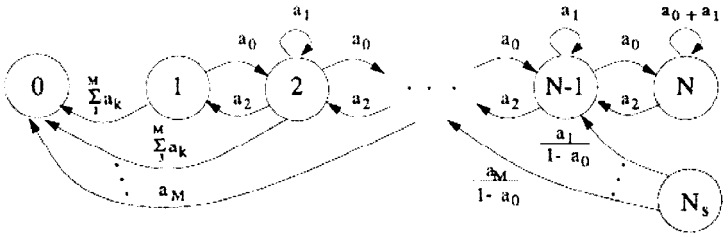


Abb. 5 Markovkette im Falle der verallgemeinerte gepulsten Entladung [7]

Durch Experimente auf der Grundlage der beiden Modelle erkannte man, dass sich die Batterieperformance enorm anheben lässt, wenn sich Wahrscheinlichkeit für den Recovery-Effekt erhöhen. Des Weiteren muss der Initialzustand N_s größer sein als die maximale Burstgröße, um eine effektive Energieversorgung während eines Packetburstes sicherstellen zu können. Zuletzt sei hier noch anzumerken, dass eine Hinzunahme von Traffic-control Techniken in dieser Art von Kommunikationsprotokollen, sowie ein effizientes Batteriemangement die Lebenszeit der Batterie noch weiter erhöhen würde. Als Nachteilig kann man die Vernachlässigung der Passivierungseffekte sehen, sowie das nur die Stromkosten für die Kommunikation des Systems berücksichtigt werden. So genannte Hintergrundströme werden, wie oben schon bereits erwähnt, komplett vernachlässigt. Allerdings handelt es sich hier um ein stochastisches Modell, bei dem der Erfolg der Ergebnisse von der Wahl der einzelnen Übergangswahrscheinlichkeiten abhängt. Jedoch zeigt dieses stochastische Modell, welchen enormen Stellenwert die Berücksichtigung von Recovery-Effekten im Bezug auf die Maximierung der Batterielebenszeit einnimmt.

3.1.2.2 Frquency-Scaling

Während man sich in [7] nur mit den Auswirkungen von gepulster Entladung auf Kommunikationsebene auseinandersetzt, stellen Lahiri et al. in [11] mehrere Ansätze für die Architektur von effizienten Batteriesystemen vor. Auf Systemebene stellt der Einsatz von energiesparenden Konzepten besondere Anforderungen, da man hier nicht nur einzelne Komponenten hin auf ihre Energieeffizienz untersucht, sondern nun das komplette Zusammenspiel aller Komponenten betrachten muss. Diese Sichtweise beinhaltet sowohl die Hardwareebene, als auch die Softwareebene. Energiesparmechanismen müssen hier auf der einen Seite die Batterielebenszeit verlängern, den Energieverbrauch minimieren, aber auf der anderen Seite auch darauf achten, dass die Systemperformance dadurch nicht eingeschränkt wird. Ein viel versprechender Ansatz stellt hier das „Frequency-Scaling“ dar. Die eigentliche Idee ist hier, die CPU-Frequenz der aktuellen Arbeitslast anzupassen, um nicht bei geringer Arbeitslast die CPU auf hoher Frequenz zu takten. Letztendlich verbraucht eine CPU mit hoher Frequenz mehr Strom, als eine CPU mit niedriger Frequenz. Die Berechnung der Frequenz für ein Zeitintervall erfolgt nach einer „history-based-policy“, bei der der prozentuale Anteil von Arbeits- und Ruhezeit aus dem vorhergehenden Zeitintervall, sowie der Anteil der Arbeit, die im vorhergehenden Intervall nicht erledigt wurde, aufgrund einer zu niedrigen Frequenztaktung, in die Berechnung der neuen Frequenz mit einbezogen wird. Sollte nun der Anteil der Ruhezeit zu groß sein, wird die Frequenz für das kommende Intervall nach unten gesetzt, allerdings nur bis zu einer unteren Grenze, um noch die Systemstabilität zu gewährleisten. Genauso wird bei einer erhöhten Arbeitszeit die Frequenz bis hin zum Maximum erhöht. Grundsätzlich ist der Ansatz der Frequenzskalierung sehr effizient, da man zum einen die Frequenz so regeln kann, dass sie das aktuelle Batteriemodell berücksichtigt (Rate-Capacity Effekte, Recovery-Effekte, Alterungseinflüsse) und zum anderen durch die Verwendung von Workload-Charakteristiken, Performancerestriktionen und den Energiebedarf einzelner Systemkomponenten ein Gleichgewicht zwischen Energieeffizienz und Performance herstellen kann. Allerdings kann es passieren, dass bei sehr komplexen Systemen und hochgradig dynamischen Workloads zwar Energie gespart wird, aber die Performance drastisch abnimmt, was bei Echtzeitsystemen das Überschreiten von Deadlines nach sich ziehen kann. Hier muss in einem größeren Anwendungskontext abgewogen werden.

3.1.2.3 Dynamic Power Management

Alle bisher vorgestellten Ansätze beziehen zwar charakteristische Eigenschaften einer Batterie, wie beispielsweise den Recovery-Effekt, in ihre Berechnungen mit rein, jedoch werden keine Möglichkeiten angeboten, wie man Systeme auf den aktuellen Ladestand der Batterie anpassen kann. Sich mit diesem Problem auseinander zu setzten haben sich L. Benini in [10] zur Aufgabe gemacht. Hier wird das Prinzip des „Dynamic Power Management" am Beispiel eines digitalen Audio-Recorders vorgestellt. Grundsätzlich versucht man mit dem Arbeiten in verschiedenen Systemzuständen, je nach Batteriestatus, einen respektablen Kompromiss zwischen Systemperformance und Batteriestatus zu erreichen. In [10] stehen 5 verschiedene Zustände zur Verfügung (off, sleep, idle, raw-sound und fine-sound). Der Zustand „fine-sound" repräsentiert hierbei einen qualitativ hochwertigen Musikmodus, während der Zustand „raw-sound" einen qualitativ minderwertigen Musikmodus darstellt. Entscheidend sind für die jeweiligen Zustandsübergänge die Energiekosten, um von einem Zustand zum anderen zu wechseln (Abb. 6), sowie die Verzögerungskosten (Abb. 7) der jeweiligen Übergänge. Für die Umsetzung einer dynamischen Strategie stehen zwei unterschiedliche Mechanismen zur Verfügung. Zum einen eine „open-loop" Politik (time-out Politik), bei der vom „fine-sound-Zustand" in den Zustand „idle" gewechselt wird, sobald keine Musik mehr gespielt wird. Nun bleibt das System so lange in diesem Zustand, bis ein erster Time-out abgelaufen ist. Nach Ablauf dieser Schranke wechselt das System in den Zustand „sleep", in dem so lange verweilt wird, bis auch eine zweite Zeitschranke abgelaufen ist und den Übergang zum Zustand „off" ermöglicht. Sollte jedoch während einer der beiden Zeitschranken wieder Musik abgespielt werden, so wird wieder in den „fine-sound-Zustand" gewechselt. Bei diesem Mechanismus geht es nur darum den Stromverbrauch durch den Wechsel in unterschiedliche Ruhezustände zu reduzieren, aber auch die Möglichkeit zu bieten mit wenig zusätzlichem Energieaufwand wieder in den aktiven Zustand zu wechseln. Der zweite Mechanismus nimmt hingegen auch die Ausgangsspannung der Batterie mit in seine Entscheidung auf. Hier wird eine „closed-loop" Politik (Spannungsgrenzwert Politik) zur Verfügung gestellt, bei der das System vom „fine-sound-Zustand" in den qualitativ niedrigeren „raw-sound-Zustand" wechselt, wenn die Batteriespannung unter einen festgelegten Grenzwert (Threshold V_{th}) fällt. Für die Berechnung des Grenzwertes gelten folgende Zusammenhänge:

1. Qualitätsfaktor Q wird nach (1) berechnet, welcher den prozentualen Anteil der gespielten Zeit im qualitativ hochwertigen Zustand im Vergleich zur Gesamten Spielzeit angibt.

$$Q = T_{Fine} / (T_{Fine} + T_{Raw}) \quad (\mathbf{1})$$

2. Metrik P wird nach (2) erstellt, die sich aus der normalisierten Batterielebenszeit (NLT) und Q zusammensetzt.

$$P = NLT \times Q \quad (\mathbf{2})$$

3. Der optimale Grenzwert V_{th} ist derjenige, der P maximiert

Eine Kombination beider Mechanismen bringt hier einen maximalen Erfolg. So ergaben genauere Untersuchungen, dass mit der „open-loop" Politik alleine, eine Verbesserung um ca. 14 % erhält. Mit der closed-loop Politik alleine sind Verbesserungen von bis zu 22 % möglich und eine Kombination beider Politiken ergab eine Verbesserung von bis zu 66 %. Dies ist zum einen auf die abgestuften Ruhezustände zurück zu führen, wenn der Audio-Recorder im qualitativ hochwertigen Modus spielt und somit der Batterie die Möglichkeit der Erholung gibt (Recovery-Effekt). Zum anderen, der eintretende qualitativ minderwertige Modus, wenn V_{th} erreicht wurde, bis die Batterie komplett entladen ist. Jedoch ist die Wahl der Grenzwerte im ersten Mechanismus sehr schwierig, da die Übergänge von „spielen" zu „pause" sehr benutzerabhängig sind und somit die Wahl von fixen Grenzwerten nicht Vorteilhaft ist. Des Weiteren kommt der Wahl von V_{th} eine entscheidende Rolle zu. Sollte die Grenzspannung zu hoch angesetzt werden, so kann es passieren, dass der Recorder zu früh in den „raw-sound-Zustand" wechselt und somit den Benutzer verärgert, da dieser evtl. einen besseren Sound einer längeren Lebenszeit vorzieht. Wird er aber zu niedrig angesetzt, so kann es passieren, dass die Batterie schon so verbraucht ist, dass der „raw-sound-Modus" keine signifikante Verlängerung der Laufzeit herbeiführen kann.

Table 1. Energy cost per state transition.

	RawSound (mJ)	FineSound (mJ)
Off	49.5	99.0
Sleep	11.5	29.7
Idle	2.0	4.1

Abb. 6 Energiekosten

Table 2 Latency cost per state transition

	RawSound (mJ)	FineSound (mJ)
Off	150	200
Sleep	70	100
Idle	40	60

Abb. 7 Verzögerungskosten

3.1.3 Bewertung der Batterie-zentrischen Ansätze

Die hier vorgestellten Anätze unterscheiden sich primär in der Anzahl zu Verfügung stehender Batterien. Im Bereich der Mehr-Batterie-Systeme wurden zwei Möglichkeiten vorgestellt. Zum einen die hochfrequente Alternierung zwischen heterogenen Batterien und zum anderen das „Terminal-Voltage-Based-Scheduling“. Beide Ansätze verfolgen primär das Ziel, eine Batterie mit doppelter Kapazität durch hochfrequente Alternierung zwischen den Batterien zu simulieren (wenn man ein Mehr-Batterie-System mit zwei Batterien betrachtet). Beide verwenden eine Round-Robin-Politik, bei der jede Batterie für eine bestimmte Dauer einen variablen Energiebetrag abgeben muss, abhängig von der jeweiligen Restkapazität. Um jedoch den abzugebenden Energiebedarf berechnen zu können, muss im Voraus die prozentuale Verteilung der Stromstärken, die das System benötigt, bekannt sein. Die Berechnung des Round-Robin-Schedules muss allerdings zur Laufzeit getätigt werden, da die abzugebende Energie jeder Batterie von der Restkapazität abhängt. Der Unterschied zwischen beiden Varianten ist der, das beim „Terminal-Voltage-Based-Scheduling“ die Batterien dynamisch aus dem Set der zur Verfügung stehenden Batterien heraus genommen werden, wenn deren Ausgangsspannung unter einen gegebenen Grenzwert fällt. Somit haben diese Batterien die Möglichkeit, sich über einen längeren Zeitraum zu erholen. Beide Ansätze haben aber ein sehr großes Sparpotential, wenn man zum einen heterogene Batterien verwendet und zum anderen das System auch in unterschiedlichen Spannungsebenen arbeitet. Nachteilig bei der zweiten Variante ist zu bewerten, dass das Herausnehmen von Batterien aus der Round-Robin-Politik nur bei einem Set von sehr vielen Batterien einen Sinn macht, da sonst im Falle eines Zwei-Batterie-System nur noch eine Batterie vorhanden ist, die die komplette Systemenergie liefern muss.

Ebenfalls wurden zwei Möglichkeiten im Bereich der Ein-Batterie-Systeme näher erläutert. Jedoch verfolgen beide Ansätze unterschiedliche Strategien. Im Falle der gepulsten Entladung will man die Batterie für kurze Zeit belasten und ihr danach die Möglichkeit der Erholung bieten. Der Vorteil dieses Ansatzes ist darin zu sehen, dass die Batterie während ihrer Erholungsphase Recovery-Effekte ausnutzen kann und somit zusätzlich verlorene Energie durch hochohmige Entladung wieder zurück gewinnen kann. Allerdings liefert die gepulste Entladung nur dann gute Ergebnisse, wenn die Batterie auch tatsächlich die Möglichkeit der Erholung bekommt, was bei voll ausgelasteten Systemen nicht der Fall sein dürfte. Die zweite vorgestellte Variante befasste sich mit dem dynamischen Power Management. Das Arbeiten in unterschiedlichen Systemzuständen, abhängig von der Restkapazität der Batterie oder des Benutzerverhaltens stand hier im Vordergrund. Zum einen wurde einer Time-out-Politik betrachtet, die es dem System ermöglicht abhängig vom Benutzerverhalten in unterschiedlich abgestufte Ruhezustände über zu gehen. Zum anderen war es mit der Spannungsgrenzwert-Politik möglich, mit qualitativ abgestuften Systemzuständen auf die aktuelle Restkapazität der Batterie zu reagieren. Jede Politik für sich betrachtet liefert je nach Anwendungsfall zwar eine Verlängerung der Lebenszeit, jedoch ist diese viel geringer, als bei einer Kombination beider Politiken. Der zusätzliche Aufwand hält sich beim dynamischen Power-Management in einem überschaubaren Rahmen. Lediglich die Ausgangsspannung der Batterie muss gemessen werden, da die Übergänge in unterschiedliche Ruhestadien nur von fixen Grenzwerten abhängt. Allerdings ist die Wahl der Grenzwerte entscheidend für den Erfolg dieser Strategie, denn setzt man sie zu niedrig an kann es passieren, das man den „off-Zustand“ zu früh erreicht und der Benutzer unnötig lange auf seine Musik warten muss. Außerdem kommt der Wahl des Spannungsgrenzwertes eine entscheidende Rolle zu, bei der ein zu später Wechsel in einen qualitativ minderwertigen Modus, die Batterie schon so verbraucht ist, das dieser Modus die Lebenszeit nicht erheblich verlängert.

Tabelle 1 enthält nochmals eine Übersicht über alle vorgestellten Ansätze, sowie ihre jeweilige Bewertung nach relevanten Kriterien.

	Fast Alternation	**Terminal Voltage Scheduling**	**Pulsed Discharge**	**Dynamic Power Management**
Batterie-System	Mehr-Batterie-System	Mehr-Batterie-System	Ein-Batterie-System	Ein-Batterie-System
Energie sparen durch:	hochfrequente Alternierung zwischen Batterien	dynamische Variante der Alternierung	Gepulste Entladung	alternative Systemzustände
Sparpotential	sehr gut, wenn heterogene Batterien verwendet werden	sehr gut, bei einer großer Anzahl von Batterien	gut, wenn die Batterie tatsächlich Erholungsphasen bekommt	sehr gut, bei einer Kombination beider Politiken
Zusätzlicher Aufwand	Online-Berechnung des Round-Robin-Schedules	Online-Berechnung des Round-Robin-Schedules + Erstellung des aktuellen Batteriesets	in diesem Beispiel: Erstellung eines stochastischen Zustandsmodell	Messung der Batterie-Ausgangsspannung
Vorwissen	prozentuale Verteilung der Stromstärken	prozentuale Verteilung der Stromstärken + Spannungsgrenzwert	in diesem Beispiel: Energieverbrauch pro Kommunikation und Höhe des Recovery-Effektes	Benutzerverhalten und Wahl der Grenzwerte

Tabelle 1: Übersicht über die Batterie-zentrische Ansätze

3.2 Scheduling-zentrische Ansätze

In diesem Abschnitt werden nicht mehr die einzelnen Komponenten eines Systems und ihr Energieverbrauch betrachtet, sondern man nähert sich dem Problem des Energiesparens von der Betriebssystem und Softwareseite her. Jedoch setzt diese Herangehensweise voraus, dass man den Energieverbrauch der einzelnen Betriebssystem- und Softwarekomponenten (speziell: einzelnen Task), so gut wie möglich modellieren und vorhersagen kann. Die Vorhersage kann aber oft nicht vor dem Systemstart gegeben werden, sondern oftmals muss aufgrund der Komplexität und dynamischer Schedules, eine Berechnung während der Laufzeit durchgeführt werden.

Der Hauptenergieverbrauch in vielen Systemen stellt das Betriebssystem dar, da hier auch die Schnittstelle zur eingesetzten Hardware verankert ist und diese somit, auch den Energieverbrauch der Hardware steuert (on, off, idle, busy). Des Weiteren übernimmt das Betriebssystem auch Aufgaben wie Scheduling, Memory-Management, Kommunikation, bei denen es sich lohnt energiesparende Konzepte einzusetzen. In der Vergangenheit haben sich viele Wissenschaftler mit der Modellierung des Energieverbrauchs von Software beschäftigt. Diese Ansätze wurden von T. Li in [12] in folgende Kategorien unterteilen:

Instruction Level Power modelling
Grundgedanke hier, ist die verbrauchte Energie mit einem individuellen Befehl in Verbindung zu bringen. [9] hat sich eingehend mit dieser Problematik beschäftigt. Jedoch ist die Modellierung einzelner Befehle in Verbindung mit ihrem Energieverbrauch nur für einfache und nicht zu komplexe Software möglich, da bei großen Anwendungen oftmals unvorhergesehene Sprünge eintreten oder die Befehlabfolge einfach zu komplex ist, was den Speicherplatz für eine Modellierung sprengen würde.

Characterization-based macro-modeling
Hier entfernt man sich von der Modellierung einzelner Befehle, und sieht ganze Funktionen bzw. Subroutinen einer Anwendung als „black boxes“ an und fasst somit viele einzelne Befehle zu einem Paket zusammen. Den Energieverbrauch berechnet man jetzt nur noch mit den Eigenschaften, die für diese Box von Interesse sind. Allerdings kommt hier nur eine offline Berechnung in Frage, da das Finden von Blöcken, die man zusammenfassen kann, aufgrund von Programmabhängigkeiten sehr zeitaufwendig ist

Performance counter-based run-time power estimation
Dieser Ansatz baut auf der Annahme auf, dass die Energie, die durch die Ausführung von Software verbraucht wird mit der Anzahl von Zugriffen und Wechseln innerhalb der Prozessoreinheit in Verbindung gebracht werden kann. Um diese Anzahl der Zugriffe zu zählen, sind viele modere Mikroprozessoren mit „event-countern“ ausgestattet, mit deren Hilfe es möglich ist Heuristiken über energierelevante Events zu erstellen. Auf Grund dessen, das in heutiger Zeit viele Befehle parallel abgearbeitet werden, ist hier die Anzahl der Events, die man parallel messen kann, entscheidend für die Genauigkeit der Berechnung.

3.2.1 Power-aware Scheduling

Aufgrund der aufgezeigten Möglichkeiten, zur Messung des Energiebedarfs einzelner Befehle oder ganzen Tasks, ist man nun in der Lage geeignete energiesparende Schedules für Pragrammabläufe zu erstellen. So stellen J. Liu et al. in [13] ein „power-aware-Scheduling“ unter Zeitbedingungen vor, welches hauptsächlich für missionskritische eingebettete Systeme entwickelt wurde. Hier wurde für die Modellierung des Energieverbrauchs von Tasks das „Characterization-based macro-modeling“ verwendet. Am Beispiel des Mars Pathfinders wird die Funktionsweise dieses Schedules näher erläutert. „Power-aware“ bedeutet in diesem Zusammenhang, die zur Verfügung stehende Energie optimal auszunutzen, was nicht immer dem Konzept des „low-power-design“ entsprechen muss. Man geht hier von dem Ansatz aus, das man die Umsetzung des „power-aware-designs“ nicht nur auf der Ebene der Komponenten umsetzen darf, sondern auch auf der Systemebene, um die Verteilung der zu Verfügung stehender Energie optimal unter den Komponenten aufzuteilen. Die reine Verwendung von energiesparenden Mechanismen auf Komponentenebene haben den Nachteil, dass sie keine Zeitbedingungen wie beispielsweise Deadlines oder Abhängigkeiten der Komponenten untereinander berücksichtigen können. Des Weiteren unterscheiden sie nicht zwischen freier Energie (Solarenergie), die wenn sie nicht verbraucht wird sinnlos verschwendet wird, und teurer Energie (nicht widerauflabbare Batterien). Demnach werden abzuarbeitende „worksets“ als gegeben hingenommen und man minimiert gemäß dem „low-power-design“ nur den Energieverbrauch der einzelnen Komponenten. Der hier entwickelte „power-aware“ Scheduler beachtet minimale und maximale Zeitbedingungen, sowie das maximal zur Verfügung stehende Energiebudget (freie Energie wird immer zuerst verbraucht). Das maximale Energiebudget setzt sich bei dieser Betrachtung aus Solarenergie und einer nicht widerauflabbaren Batterie zusammen. In Abb. 9 sind die Zeitbedingungen und in Abb. 10 der Energieverbrauch der einzelnen Mars Rover Komponenten zu sehen, die die Grundlage für einen Abhängigkeitsgraph (Abb. 11) bilden, der dann wiederum als Eingabe für den Scheduling-Algorithmus dient.

Operation	Duration (s)	Timing constraints
Heating steering motors	5	At least 5s, at most 50s before steering
Heating wheel motors	5	At least 5s, at most 50s before driving
Hazard detection	10	At least 10s before steering
Steering	5	At least 5s before driving
Driving	10	At least 10s before next hazard detection

Abb. 9 Zeitbedingungen des Mars-Rover

Power sources & loads	Duration (s)	Power (W)		
		Best case @-40 °C	Typical case @-60°C	Worst case @-80°C
20-W solar panel		14.7	12	9
Battery peak		10 max	10 max	10 max
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

Abb. 10 Energieverbrauch der einzelnen Komponenten

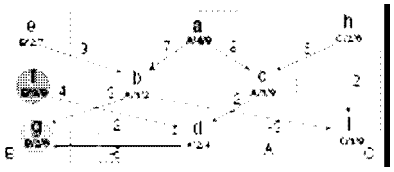


Abb. 11 Abhängigkeitsgraph

Die Aufgabe besteht nun darin, einen Schedule zu erstellen, welcher im Bezug auf die Zeitbedingungen und den Energievorrat gültig ist und die zu verbrauchende Energie minimiert. Das finden eines gültigen Schedules wird in drei Schritte unterteilt:

1) Timing Scheduling
   Auf Basis des Abhängigkeitsgraphen wird nun ein Schedule gesucht, welches von den Zeitbedingungen her gültig ist. Es kann sichergestellt werden, dass dieser Algorithmus ein Schedule findet, wenn eines existiert. Jedoch ist dieses Schedule noch nicht optimal, da das maximal zur Verfügung stehende Energiebudget noch überschritten werden kann.
2) Max Power Scheduling
   Hier wird auf Grundlage des Schedules aus 1) nach Überschreitungen des maximalen Energiebudgets zu einem Zeitpunkt t gesucht. Sollte eine Überschreitung gefunden werden, werden zu diesem Zeitpunkt Ruhepausen eingesetzt und die aktuellen Tasks verzögert ausgeführt (evtl. reicht es, nur eine Task nach hinten zu verschieben), bis man sich wieder unter der gegebenen Grenze befindet. Abb.12 zeigt ein gültiges Max-Power-Schedule. Allerdings kann nicht sichergestellt werden, dass hier ein Schedule gefunden wird, wenn eines existiert
3) Min Power Scheduling
   Hier wird eine Neuordnung des Schedules aus 2) vorgenommen, um die frei zur Verfügung stehende Energie aus dem Solarpanel voll auszulasten. Es werden aber alle Zeitbedingungen und das maximale Energiebudget beibehalten. Sinn und Zweck dieses Schrittes ist die nicht wiederaufladbare Batterie zu schonen und so viel wie möglich auf die Solarenergie auszulagern. Abb. 13 zeigt ein gültiges Schedule nach Ablauf des kompletten Schedule-Algorithmus

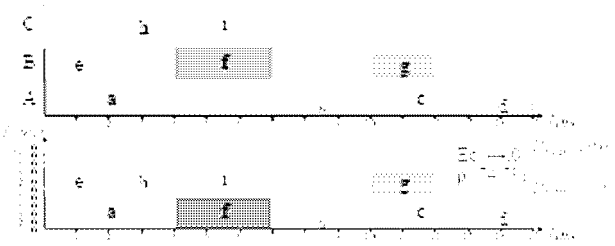


Abb. 12 gültiges Schedule nach Max-Power-Scheduling [13]

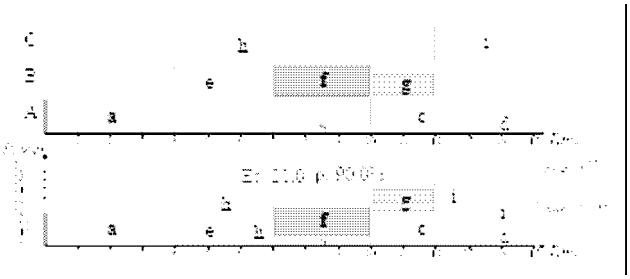


Abb. 13 gültiges Schedule nach Min-Power-Scheduling [13]

Dieser Algorithmus bietet eine optimale Möglichkeit, um gegebene Restriktionen in Zeit und Energie einzuhalten. Optimale Energieauslastung bedeutet hier, das man immer wenn es möglich ist, auf die freie Solarenergie zurückgreift. Jedoch wurden hier nur einzelne Tasks betrachtet, die zu einem Zeitpunkt beginnen und zu einem anderen wieder beendet werden. Hintergrundströme, sowie der Stromverbrauch für die Kommunikation zwischen Mars-Rover und Bodenstation werden nicht betrachtet. Dieser konstante Stromverbrauch würde dann nämlich das maximal zur Verfügung stehende Energiebudget um einiges schmälern. Des Weiteren werden die Energiekosten für die Berechnung dieser Schedules nicht beleuchtet. Diese Kosten dürften nicht minimal sein, da es bei obigem Algorithmus sehr lange dauern kann, bis ein gültiges Schedule gefunden wird. Allerdings zeigt dieses Beispiel, das man mit regenerativen Energien oder wie hier ausgedrückt mit freien Energien, die Laufzeit eines Systems erheblich verlängern kann und der teuren Energie immer wieder die Möglichkeit geben kann, durch Ausnutzung des Recovery-Effektes, die Belastung zu reduzieren.

3.2.2 Voltage/Clock-Scaling

Einen weiteren Scheduling-zentrischen Ansatz stellen D. Rakhmatov et al. In [14] vor. Hier geht man aber von einem Ein-Batteriesystem aus. Im Mittelpunkt der Betrachtung, steht auch hier das finden eines Schedules zur optimalen Energieausnutzung, allerdings mit der Möglichkeit des „voltage/clock scaling“. Man verwendet hier analytisches Batteriemodell, das zum einen sehr genau die Batterielebenszeit unter verschiedenen Entladebedingungen vorhersagen kann. Des Weiteren stellt dieses Modell einen analytischen Ausdruck zu Verfügung stellt, den man als Kostenfunktion (abgegebene Kapazität während der Dauer T) zur Verlängerung der Lebenszeit verwenden kann. Ziel ist es, diese Funktion zu minimieren, um so die Lebenszeit der Batterie zu maximieren. Um die Vorteile des Voltage/Clock-Scaling hervorzuheben, wird einmal ein Scheduling-Algorithmus ohne Voltage/Clock-Scaling vorgestellt und danach ein Algorithmus mit diesem Mechanismus. Beide Varianten liegt eine „task-load“ von acht Tasks zugrunde, die sowohl Deadlines der einzelnen Tasks, als auch Abhängigkeiten der Tasks untereinander berücksichtigt. Diese „task-load“ dient beides Mal als Eingabe für

die Berechnung eines energie-effizienten Schedules. Des Weiteren sind auch hier für jede Task die Ausführungszeit und der Energieverbrauch bekannt

3.2.2.1 Task sequencing without scaling
Hier findet eine Minimierung der Kostenfunktion unter folgenden Bedingungen statt. Alle Taskabhängigkeiten müssen erhalten bleiben (Bedingung1: dependency constraint), ein vorgegebenes Verzögerungsbudget darf nicht überschritten werden (Bedingung2: delay constraint) und die Batterie muss die ganze Zeit über die geforderte Kapazität liefen können (Bedingung 3: endurance constraint). Es wird lediglich ein Schedule erstellt, das die Startzeiten der einzelnen Tasks enthält. Die Erstellung erfolgt in drei Schritten:

1) Greedy sequencing
   Dient dem Erstellen eines Schedules unter Berücksichtigung der Abhängigkeiten. Das Einfügen von Ruhepausen ist hier nicht erlaubt. Um Bedingung 2 (delay constraint) einzuhalten, wird das Budget B so gesetzt, dass es größer oder gleich der Gesamtausführungszeit alle Tasks ist.

2) Incremental recovering
   In diesem Schritt wird sicher gestellt, dass Bedingung 3 eingehalten wird, indem man an den geeigneten Stellen Ruhepausen (idle periods) so einsetzt, dass das Taskset aus i) ohne Unterbrechung durchlaufen kann. Die Ablaufreihenfolge wird aber nicht geändert.

3) Local compressing
   Durch das Einfügen von Ruheperioden in ii) kann es nun aber passieren, dass das Verzögerungsbudget überschritten wird. Es werden Tasks mit geringen Energieanforderungen (light loads) in die idle-Perioden unter Berücksichtigung der Abhängigkeiten gelegt. Somit wird auch Bedingung 2 wieder eingehalten.

3.2.2.2 Task sequencing with voltage/clock scaling
Sinn und Zweck ist es die benötigte Spannung und Frequenz in einem System so niedrig wie möglich zu halten, um einer zu starken Belastung der Batterie vorzubeugen. Jedoch muss man sich darüber im Klaren sein, das sich eine Spannung- und Frequenzänderung auf die Stromstärke und Ausführungszeit der Tasks auswirkt. Somit bewirkt eine Änderung der Spannung ebenfalls eine Verringerung der Stromstärke, allerdings verlängert sich die Ausführungszeit. Des Weiteren müssen die gleichen Bedingungen wie oben schon erwähnt eingehalten werden. Um nun ein passendes Schedule zu finden verwendet man den Ansatz des „scaling based on lowest-power initial solution“. Voraussetzung für diesen Ansatz ist, dass das System in verschiedenen Spannungsleveln arbeiten kann und eine Tabelle existiert, in der für jede Task, die im Bezug auf die aktuelle Spannung und Frequenz, die daraus neue resultierende Stromstärke und Ausführungszeit beinhaltet.
Nun fängt man mit der niedrigsten möglichen Spannung an und berechnet die neue Gesamtausführungszeit. Die kleinste Spannung, die Bedingung 3 erfüllt, ist nun die Spannung mit der das Schedule ausgeführt wird. Nun wird wieder mit dem schon bekannten „greedy sequencing“ ein Schedule erstellt. Das Schedule wird akzeptiert, wenn Bedingung 2 und 3 erfüllt sind. Sollte aber die Batterie nicht die geforderte Kapazität (Bedingung 3) liefern können, muss ii) und iii) angewendet werden. Wird allerdings das Verzögerungsbudget überschritten und überschreitet somit geforderte Zeitschranken, dann müssen einige Tasks auf ein höheres Spannungs- und Frequenzlevel gebracht werden, um die Ausführungszeit zu verkürzen.
Der Ansatz des Voltage/Clock-Scaling beugt zum einen einer hochohmigen Entladung der Batterie vor, da geringe Spannung auch eine geringe Stromstärke bewirkt. Zum anderen kann man durch das Einfügen von Ruhepausen, insofern keine Zeitbedingungen verletzt werden, die schon oftmals erwähnten Recovery-Effekte zu Nutze machen. Allerdings werden nur dann gute Resultate erzielt, wenn das Verzögerungsbudget hinreichend groß ist. Beispielsweise benötigt eine Task, bei einer Spannung V mit einer Stromstärke von 1000mA und einer Ausführungszeit von 5 min, bei einer Halbierung von V jetzt nur noch 125 mA, aber eine Ausführungszeit von 10 min.

3.2.3 Bewertung der Scheduling-zentrischen Ansätze
Im Bereich der Scheduling-zentrischen Ansätze wurden zwei Varianten vorgestellt. Zum einen ein Power-aware Scheduling und zum anderen das Voltage/Clock Scaling. Das Power-aware Scheduling geht von freier Energie (Solarenergie) und von teurer Energie (nicht wideraufladbare Batterie) aus. Ziel ist es die frei zu Verfügung stehende Energie so effizient wie möglich zu nutzen und die teure Energie so gut wie möglich zu erhalten. Um dieses Ziel zu erreichen wurde ein drei-stufiger Scheduling-Algorithmus vorgestellt. Das Sparpotential ist sehr groß, wenn ein existierendes Schedule auch von dem Algorithmus gefunden wird. Die Vorteile sind bei diesem Scheduling, dass man in der Lage ist, Zeitschranken und Abhängigkeiten der Tasks zu berücksichtigen, was bei Batterie-zentrischen Ansätzen nicht möglich war. Des Weiteren werden in diese Betrachtung regenerative Energien mit aufgenommen, die in der Lage sind die Systemlebenszeit signifikant zu erhöhen. Als Nachteilig ist hier zu bewerten, das die Kommunikation nicht berücksichtigt wurde, welche oftmals während der gesamten

Systemlaufzeit Energie auf hohem Niveau benötigt, egal ob gesendet/empfangen oder gewartet wird und somit würde die zur Verfügung stehende Energie um einen konstanten Faktor gesenkt werden. Im Gegensatz dazu wurde das „Voltage/Clock-Scaling“ vorgestellt. Verfolgtes Ziel ist hier die Spannung und Frequenz so gering wie möglich zu halten. Allerdings zieht eine niedrigere Spannung eine Verlängerung der Ausführungszeit nach sich. Der Vorteil dieses Verfahrens ist darin zu sehen, dass man durch die niedrige Spannung einer hochohmigen Entladung der Batterie vorbeugt. Es werden sehr gute Resultate erzielt, wenn die Deadlines der einzelnen Tasks nicht zu eng sind, da es durch die Verlängerung der Ausführungszeit oftmals zu Überschreitungen dieser Deadlines kommen kann. Bei beiden Ansätzen muss man allerdings als Vorwissen voraus setzen, das man den Energieverbrauch und die Laufzeiten der einzelnen Task kennen muss. Jedoch können diese Messungen bei einem hochgradig dynamischen System sehr umfangreich, wenn nicht gar unmöglich sein. In Tabelle 2 sind noch mal beide Verfahren und ihre Vorzüge bzw. Anforderungen zu ersehen.

	Power-aware Scheduling	**Voltage/Clock-Scaling**
Batterie-System	Mehr-Batterie-System	Ein-Batterie-System
Energie sparen durch:	Power-aware Scheduling	Voltage/Clock-Scaling
Sparpotential	sehr groß, wenn ein existierendes Schedule auch gefunden wird	sehr groß, wenn Zeitbedingungen nicht zu eng
Zusätzlicher Aufwand	Berechnung des Schedules	Berechnung des Schedules
Vorwissen	Laufzeiten und Energieverbrauch der Tasks, sowie deren Abhängigkeiten	Laufzeiten und Energieverbrauch der Tasks, sowie deren Abhängigkeiten
Vorteile	Zeitschranken, Abhängigkeiten	Vorbeugung hochohmiger Entladung
Nachteile	Kommunikation wird nicht betrachtet	Verlängerung der Ausführungszeit

Tabelle2: Übersicht über die Scheduling-zentrischen Anätze

3.3 Duty-cycle Ansätze

Diese Ansätze bieten im Gegensatz zu den Batterie- und Scheduling-zentrischen-Ansätzen einem System, dass sich in einem aktiven Zustand befindet und keine Aufgaben zu erledigen hat, die Möglichkeit sich schlafen zu legen, um somit nicht achtlos Energie im aktiven Zustand zu vergeuden. Das System soll erst dann wieder in denn aktiven Zustand übergehen, wenn wieder Aufgaben zur Abarbeitung anstehen. Um hier jedoch effizient Energie sparen zu können, sollte der Schlafzyklus länger als der Arbeitszyklus sein. Haupteinsatzgebiet der Duty-cycle-Ansätze sind Sensornetzwerke, die meist große Gebiete über einen langen Zeitraum überwachen müssen. Aufgrund der großen Anzahl verwendeter Sensor und deren Einsatz in Umgebungen mit hohem Gefahrenpotential ist ein Austausch dieser, meist zu teuer oder gar unmöglich. Deshalb ist die Maximierung der Lebenszeit des gesamten Netzwerkes eine Hauptaufgabe bei der Konzeption solcher Netzwerke. Ziel bei Sensornetzwerken ist es, so viele Sensoren im Netzwerk wie möglich auszuschalten oder in einen Ruhemodus zu versetzen, aber gleichzeitig ein Höchstmass an Netzabdeckung zu gewährleisten. Jedoch kann die Abschaltung einzelner Knoten dazu führen, das Events, die in diesem Bereich auftreten, nicht (im Falle partieller Netzabdeckung) oder erst mit großer Verzögerung (Sensor befindet sich in Ruhephase) entdeckt werden. Des weiteren kann es passieren, das Sensoren die aufgrund der Netzwerktopologie außerhalb der Reichweite der Basisstation sind, ihre Nachricht über einen erkannten Event über Nachbarknoten an die Basisstation schicken müssen. Problematisch wird dieser Versand, wenn zufällig die Nachbarknoten gerade schlafen, und somit die Basisstation keine rechtzeitige Meldung erhält. Außerdem kann es hierbei passieren, das einige Sensorknoten mehr Energie verbrauchen als andere. Diese Problematik tritt bei Sensoren auf, die in direkter Verbindung mit der Basisstation stehen, da diese meist eine erhöhte Kommunikationslast zu bewältigen haben. So müssen diese Sensoren ihre eigenen erkannten Events an die Basisstation schicken, sowie auch die erkannten Events von zu weit entfernten Knoten, die keine direkte Verbindung zur Basisstation haben. In dieser Ausarbeitung gehe ich von Sensornetzwerken aus, die mehrere Sensorknoten, aber nur eine Basisstation besitzen.

[15] und [16] lösten den Konflikt zwischen Netzabdeckung und der Lebenszeitmaximierung mit einer partiellen Netzabdeckung. Hierbei werden zwei Schedules betrachtet, die es einem Sensor ermöglichen, sich schlafen zu legen. Zum einen ein Zufallsschlaf, bei dem jeder Knoten unabhängig von seinen Nachbarknoten eine Schlaf- und Weckzeit auswählt. Zum anderen der synchronisierte Schlaf, bei dem alle Knoten gemeinsam schlafen gehen und auch gemeinsam wieder aufwachen. Jedoch haben diese Ansätze den Nachteil, dass oftmals eine ausreichende Netzabdeckung nicht gewährleistet war, was dazu führte, dass Events, die das Netzwerk eigentlich erkennen und behandeln sollte, nicht erkannte und unbehandelt blieben. Allerdings führte die synchronisierte Variante zu guten Ergebnissen, wenn zu erkennende Events periodisch oder zumindest vorhersagbar auftreten.

3.3.1 Sleep Scheduling for Rare-Event Detection

Q. Cao et al. erkannten die obige Schwachstelle, gerade im Bezug auf die Erkennung seltener oder aperiodischer Ereignisse und entwickelte in [17] einen nahezu optimalen Sleep-Scheduling-Algorithmus für Sensornetzwerke, mit dem sich die durchschnittliche Erkennungszeit minimieren und die Lebenszeit des Netwerkes erhöhen lässt. Des Weiteren legte man Wert darauf, die Verzögerung der Paketversendung über Nachbarknoten zu minimieren. Der zweistufige Scheduling-Algorithmus arbeitet nach dem Prinzip der rotierenden partiellen Netzabdeckung, bei der aber sichergestellt ist, das jeder Punkt im Netzwerk innerhalb einer festen Zeitgrenze (finite delay bound) abgetastet wird.

In der ersten Stufe (primary subset) wird eine minimale Menge aller Knoten erstellt, die in der Lage sind, das komplette Gebiet zu überwachen. Alle Knoten, die nicht in dieser Knotenmenge sind, werden abgeschaltet. Um aber nicht immer die gleichen Knoten im Set zu haben, wird dieser Vorgang periodisch in langen Zeitabständen (mehrere Stunden) wiederholt, was den ruhenden Knoten die Möglichkeit einer sehr langen Erholungspause gibt. Mit der so erstellten Knotenmenge beginnt nun Schritt zwei. In Abbildung 14 wird der schematische Ablauf des nun folgenden Verfahrens, zur Findung des optimalen duty-cycle, dargestellt. Hier wird es den Knoten des „primary set“ durch das duty-cycling Verfahren ermöglicht Energie zu sparen. Prinzipiell gilt, je kürzer der Arbeitszyklus, desto mehr Zeit hat man für die Ruhepause. Vorraussetzung für diesen Schritt ist, dass alle Knoten eine synchronisierte Uhrzeit haben, und dass jeder Knoten seine Nachbarn kennt, die den gleichen Bereich abdecken (Nachbarknoten). Jeder benachbarte Knoten wählt nun unabhängig voneinander eine Aufwachzeit und sendet diese seinen Nachbarn zu. Nun beginnt für jeden Knoten nach Ablauf einer bestimmten Zeit jeweils eine neue Iteration, in der jeder versucht seine Aufwachzeit im Sinne Minimierung der Erkennungszeit eines Events zu optimieren, indem er in seine Berechnung auch die Aufweckzeiten der anderen Knoten mit einbezieht. Sollte eine Minimierung der Erkennungszeit erreicht worden sein und sollte diese im Vergleich zur vorherigen Iteration größer als ein vorgegebener Grenzwert sein, so wird die neue Aufwachzeit durch die alte ersetzt. Bleibt diese jedoch unverändert, so beginnt der Knoten mit seinem duty-cycling Prozess. Hierbei bildet die Aufwachzeit den Zeitpunkt in jedem Zyklus, an dem der Sensor für einen fest vorgegebenen Zeitraum aktiv ist und danach wieder in den Status „inaktiv“ übergeht.

Allerdings kann es bei diesem Schedule passieren, dass keine geschlossene Verbindung von einem Sensor zur Basisstation besteht, besonders wenn dieser am äußersten Rand des Netzwerkes angesiedelt ist. Dieser Umstand ist darauf zurück zu führen, dass bei einem hochfrequenten duty-cycle die Wahrscheinlichkeit sehr gering ist, dass alle Sensoren zur selben Zeit wach sind. Deshalb synchronisiert jeder Knoten sein duty-cycle mit dem Nachbarknoten, der der Basisstation am nächsten ist. Diese Technik nennt sich „streamlined weakeup“ und wird ebenfalls in [17] explizit beschrieben. Dieser Ansatz liefert zwar sehr gute Ergebnisse in der Kategorie „Maximierung der Lebenszeit eines Sensornetzwerkes“ und Erkennung seltener Ereignisse, jedoch beschränkt man sich hier nur auf die Erkennung und dem sicherstellen einer Verbindung zur Basisstation. Energiefresser wie die Kommunikationseinheit eines Sensors, zum senden und empfangen von Nachrichten werden hier nicht betrachtet. Gerade die Kommunikation zwischen Sensoren verbraucht einen Hauptteil der Energie, besonders bei denen, die zusätzliche Kommunikation von anderen Sensoren weiterleiten müssen. Hier ist es oft nicht möglich lange Ruhepausen einzufügen.

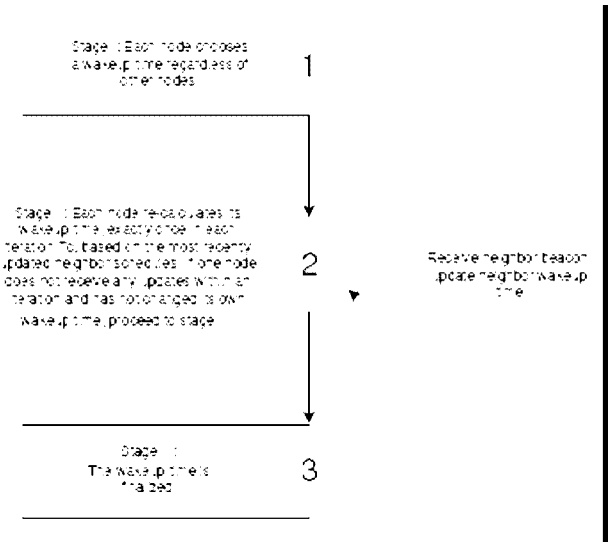


Abb. 14 State Transition of Optimization Algorithm

3.3.2 Quasi-Markov Sleep Scheduling

R. Subramanian und F. Fekri beschäftigten sich deshalb in [18] eingehend mit dem Energieverbrauch von Sensornetzwerken während der Kommunikation. Aufgrund dessen, das viele Sensoren nicht direkt ihre Daten zur Basisstation senden können, müssen sie diese über andere benachbarte Knoten, die näher an der Basis sind, senden. Dies macht es aber notwendig, dass benachbarte Sensoren, aber auch zu gleichen Zeit wach sind und die empfangenen Daten weiter leiten. Eine mögliche Lösung wurde bereits im vorherigen Abschnitt erläutert. Ein Nachteil der Synchronisierung des duty-cycle benachbarter Knoten besteht darin, dass während des Arbeitsphase das Kommunikationsmodul die ganze Zeit Energie verbraucht, egal ob Daten gesendet werden oder sich das Modul im „idle-Status" befindet und mittels „idle-listening" auf eingehende Daten warten. Diese Erkenntnis geht auf [19] [20] zurück, die herausfanden, dass idle-listening fast genau so viel Energie verbraucht wie das eigentliche Empfangen von Daten. In dieser Ausarbeitung wird ein Ansatz vorgestellt, der keine Synchronisation der Knoten benötigt. Hier entwickelt man einen asynchronen Ansatz, bei dem die Knoten unabhängig voneinander entscheiden, wann sie wach sind und wann sie sich schlafen legen (quasi-markov sleep Scheduling, Abb.15). Der Energieverbrauch des idle-listening wird dadurch minimiert, das der Sensor nach dem aufwachen prüft, ob der Sendekanal frei oder belegt ist. Sollte der Kanal belegt sein, so wechselt der Knoten mit seiner Kommunikationseinheit in einen low-power Modus. Dieser dient nur dazu, ankommende Daten in einem Puffer zu speichern und zu sendende Daten im Puffer zu behalten, bis ein Kanal frei wird, um diese zur Basis oder einem Nachbarknoten zu senden. Sollte hingegen der Kanal zu einem Nachbarknoten frei sein, werden Daten aus dem Puffer versendet und evtl. ankommende Daten durchgeleitet oder weiter verarbeitet. Auch wird hier keine Gleichverteilung der Sensoren vorgeschlagen wie in Abb. 16 zu sehen ist, sondern eine nicht-uniforme Verteilung (Abb. 17). Diesem Ansatz liegt der Aspekt zu Grunde, das in einem gleich verteilten Netzwerk die Knoten in der Nähe der Senke mit einer sehr hohen Kommunikationsbelastung konfrontiert werden, was sich letztendlich auf einen höheren Energieverbrauch und somit auf eine kürze Lebensdauer nieder schlägt.

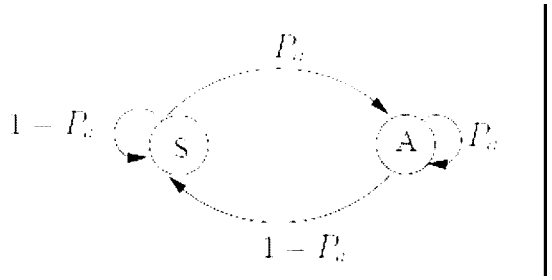


Abb. 15 Zustandsdiagramm für asynchrones Sleep-scheduling

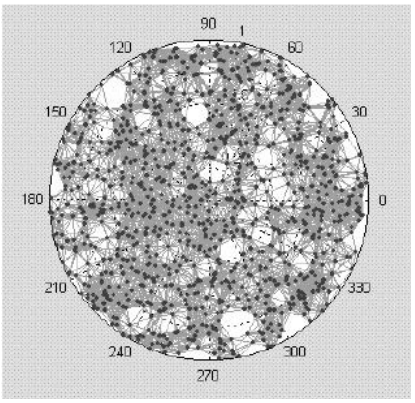


Abb. 16 gleichverteiltes Sensornetzwerk

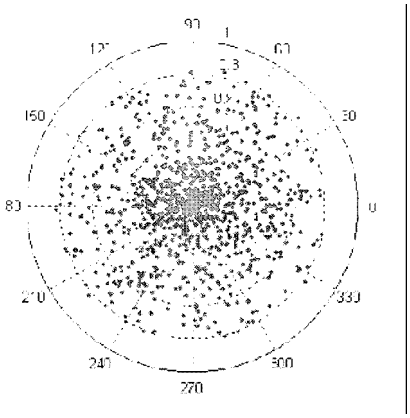


Abb. 17 nicht-gleich verteiltes Sensornetzwerk

3.3.3 Bewertung der Duty-Cycle Ansätze

Zum einen wurde ein Sleep-Scheduling vorgestellt welches sicher stellt, das jeder Punkt in einem Netzwerk innerhalb einer vorgegebenen Zeitgrenze abgetastet wird, um somit auch seltene Events erkennen zu können und gleichzeitig aber die Lebenszeit des gesamten Netzwerkes maximiert. Allerdings ist der Aufwand für die Umsetzung dieses Schedules nicht unerheblich. So muss man hier seine Nachbarknoten kennen, was gerade in dynamischen Netzwerken große Probleme bereiten kann, da sich hier die Position der einzelnen Sensorknoten oftmals ändert und somit auch deren Nachbarschaft. Des Weiteren müssen die Duty-Cycles untereinander abgestimmt werden, was einen zusätzlichen Kommunikationsoverhead zur Folge hat. Um aber die Duty-Cycles untereinander abstimmen zu können ist es hier notwendig, dass alle Knoten von einer synchronisierten Uhrzeit ausgehen. Nachteilig an diesem Ansatz ist, dass die Sensoren nur für die Erkennung der Events Energie verbrauchen. Der Energieverbrauch für die Kommunikation untereinander wird nicht betrachtet. Deshalb wurde ein zweiter asynchroner Ansatz, das „Quasi-Markov-Sleep-Scheduling“, vorgestellt, bei dem jeder Knoten seine Wach- und Ruhezeiten unabhängig von seinen Nachbarn wählt. Hier steht der Energieverbrauch bei der Kommunikation im Fordergrund. Vorteilhaft ist neben der Unabhängigkeit der Sensorknoten, das das Kommunikationsmodul einen „Low-Power-Modus“ im Aktivzustand verwendet, wenn der Kommunikationskanal zum Nachbarknoten nicht frei ist, was zusätzliche Energie einspart. In diesem Fall werden die zu sendenden Daten, sowie durch zu leitende Kommunikation von einem Nachbarknoten, in einem Puffer zwischengespeichert, bis der Kommunikationskanal wieder frei ist. Der verwendete Puffer in jedem Knoten stellt allerdings einen zusätzlich Aufwand dar. Jedoch kann es durch die asynchrone Wahl der Schlafzeiten passieren, das keine geschlossene Verbindung von einem Sensor zur Basis besteht, was evtl. eine verspätete Reaktion der Basis auf Ereignisse zur Folge haben kann. Tabelle 3 gibt noch einmal einen Überblick über die präsentierten „Duty-Cycling-Verfahren“.

	Sleep-Scheduling for Rare-Event-Detection	**Quasi-Markov Sleep Scheduling**
Optimiert für:	Erkennung seltener Events	Energieverbrauch bei Kommunikation
zusätzlicher Aufwand	Nachbarschaft erkennen, Duty-Cycle abstimmen	Puffer für ein- und ausgehende Nachrichten
Vorwissen	synchronisierte Uhrzeit, Nachbarn müssen bekannt sein	geschätztes Kommunikationsaufkommen
Vorteile	Erkennung seltener Event	Duty-Cycle nicht von Nachbarknoten abhängig
Nachteile	Kommunikation wird nicht betrachtet	evtl. verspätete Reaktion auf Events

Tabelle 3: Übersicht über die Duty-cycle Ansätze

3.4 Bewertungen aller Ansätze

Grundsätzlich benötigen alle Ansätze eine möglichst realitätsnahe Beschreibung des Batterieverhaltens, um so exakt wie möglich die reale Restkapazität der Batterie bestimmen zu können. Batterie-zentrische Ansätze haben den Vorteil, da sie direkt an der Batterie ansetzen und effektiv Recovery-Effekte ausnutzen zu können. Jedoch betrachten diese Ansätze nicht die System- und Softwareseite. Auf der Systemseite liefern jedoch die Scheduling-zentrischen Ansätze sehr gute Resultate. Bei ihnen ist es möglich Zeitschranken und Abhängigkeiten der Tasks zu berücksichtigen, um somit energie-effiziente Schedules zu erstellen. Allerdings kann es bei einer gleichzeitigen Anwendung von Batterie- und Scheduling- Ansätzen zu einer gegenseitigen Überlagerung kommen. So sind Batterie-zentrische Ansätze sehr effektiv, wenn die Batterie Ruhepausen bekommt, jedoch zielen Scheduling-zentrische Ansätze auf eine effektive Ausnutzung der zur Verfügung stehenden Energie, was nicht unbedingt das Einfügen von Ruhepausen beinhaltet. Die Anwendung beider Ansätze muss somit von Fall zu Fall neu betrachtet werden und auf die spezielle Situation abgestimmt werden. Um effektiv Energie in Sensornetzwerken zu sparen, liefern die Duty-cycle-Ansätze gute Ergebnisse. Jedoch ist bei deren Anwendung wichtig, dass man sich darüber im Klaren ist, welche Art der Überwachung (treten Events zyklisch oder azyklisch auf, wie schnell muss auf ein erkanntes Event reagiert werden) das Netzwerk übernehmen soll und wie hoch die zu erwartende Kommunikationslast eingeschätzt wird. Muss beispielsweise schnell auf erkannte Events reagiert werden, muss sicher gestellt sein, dass auch jeder Knoten eine geschlossene Verbindung zur Basis besitzt und nicht erst warten muss bis sein Nachbar wieder mit seinem Arbeitzyklus beginnt. Somit kann es in diesem Fall passieren, das einige Knoten länger aktiv bleiben müssen, obwohl sie zur Zeit keine Arbeit zu verrichten haben. Der Vorteil von Duty-cycle Ansätzen ist die Möglichkeit, die Lebenszeit ganzer Systeme, wie beispielsweise Sensornetzwerke, zu maximieren und nicht nur einzelne Subsysteme. Allerdings kommt der Anzahl verwendeter Sensoren in einem Netzwerk eine entscheidende Rolle zu. So können Duty-Cycle Ansätze nur dann gute Ergebnisse liefern, wenn auch Sensoren vorhanden sind, die sich in einem Bereich überschneiden, um sich somit die Überwachungsarbeit teilen zu können. Ein optimales Ergebnis ist bei diesen Ansätzen erreicht, wenn alle Sensoren zur gleichen Zeit ihren Dienst, auf Grund fehlender Energie versagen und somit das komplette Netzwerk seinen Dienst einstellt.

Schlussendlich kann man sagen, dass jede der drei Kategorien von Energiesparmechanismen zu einer effizienten Nutzung begrenzter Batteriekapazität beiträgt. Die vorgestellten Mechanismen sollen sich nicht gegenseitig ausschließen, sonder vielmehr ergänzen. Letztendlich hat man erst dann einen perfekten Energiesparmechanismus, wenn die Differenz zwischen der theoretischen Lebenszeit einer Batterie und der realen Lebenszeit gegen NULL verläuft. Allerdings muss man sich bei der Anwendung aller Mechanismen darüber im Klaren sein, das schon alleine die Anwendung zusätzliche Energie verbraucht und dies in die Berechnung mit eingehen muss. In Tabelle 4 sind noch einmal alle Zusammenhänge stichwortartig zusammen gefasst.

	Batterie-zentrische Ansätze	**Scheduling-zentrische Ansätze**	**Duty-cycle-Ansätze**
Grundlagen	akkurates Batteriemodell	Stromverbrauch der Tasks, Zeitschranken und Abhängigkeiten	Art der Überwachung, Kommunikationslast
Vorteile	effektive Ausnutzung von Recovery-Effekten	berücksichtigen auch Zeitschranken und Abhängigkeiten	Lebenszeit ganzer System wird maximiert
Nachteile / Probleme	betrachtet nicht die Systemseite	überlagert evtl. Batterie-zentrische Ansätze	finden von Nachbarknoten, ausreichende Anzahl von Sensoren

Tabelle 4: Gesamtübersicht

4 COBIS (Collaborative Business Items)

Die praktische Umsetzung von energiesparenden Konzepten soll nun anhand des COBIS-Projekts verdeutlicht werden. Ziel bei diesem Projekt ist es, durch den Einsatz von Sensornetzwerken, Geschäftsprozesse zu dezentralisieren und an den Ort des Geschehens auszulagern.
Die herkömmliche Modellierung und Umsetzung, sowie die Überwachung von Geschäftprozessen in Unternehmen, übernehmen meist große Back-End-Systeme. Jedoch ist bei dieser Herangehensweise nachteilig, dass die Systemsoftware, die sich mit den Geschäftsprozessen auseinander setzt, meist auf Daten aus dem ganzen Unternehmen aufbaut. Um nun den momentanen Zustand eines Geschäftsprozesse bestimmen zu können, müssen diese Daten, meist manuell, über weit verteilte Bereiche des Unternehmens gesammelt werden. Durch diese manuelle Sammlung kann es aber sehr schnell zu fehlerhaften Daten kommen. Außerdem müssen diese Daten nun zentral auf dem Back-End-System gespeichert werden, um den aktuellen Zustand des Geschäftsprozesses abzubilden. COBIS bietet nun auf der Grundlage von Sensornetzwerken einen dezentralen Service-Orientierten Ansatz an, um Geschäftprozesse direkt an den Ort des Geschehens auslagern zu können und somit den Zustand eines Prozesse enger beschreiben zu können, als dies mit herkömmlichen Back-End-Systemen der Fall ist. Somit werden Daten über Objekte und Werksprozesse, beim Objekt selbst gespeichert und verarbeitet. Daten werden nur dann an den Zentralrechner geschickt, wenn eine weitere Verarbeitung nötig ist, die von den Sensorknoten nicht selbst ausgeführt werden kann. Des Weiteren können auch Daten, die ein Knoten gesammelt hat zu einem anderen Knoten gesendet werden, der dann diese Daten weiter verarbeitet, um beispielsweise eine Gefahrensituation zu erkennen. Somit können unterschiedliche Services auf unterschiedliche Knoten ausgelagert werden, was eine Kommunikation mit dem Zentralrechner unnötig macht. Das Servicekonzept hat aber auch noch den Vorteil, dass diese in service-orientierte Systemplattformen auf dem Zentralrechner integriert werden können und somit eine realitätsnahe Echtzeitabbildung der Umwelt möglich wird.

4.1 Smart Drums

Eine Mögliche Umsetzung von Collaborative Business Items zeigt das Projekt „Smart Drums“ (intelligente Fässer). Hier werden Fässer, deren Inhalt unterschiedlichste Chemikalien sind, mit einem digitalen Clip versehen. Dieser digitale Clip stellt einen Sensorknoten dar. Ziel dieses Projektes ist die Modellierung zweier Prozesse. Zum einen soll überprüft werden, ob die Lagerkapazität eines Raumes, für die Lagerung von Chemikalien, überschritten ist und zum anderen ob sich ein Fass im Bereich eines anderen befindet, die nicht gemeinsam gelagert werden dürfen.
Grundsätzlich gilt, dass die einzelnen Sensorknoten nur eine passive Rolle während der Ausführung des Geschäftsprozesses einnehmen. Die Knoten kommunizieren nur miteinander, wenn sich andere Knoten in der Nachbarschaft befinden. In diesem Fall werden „proximity informations“ ausgetauscht, welche die Ressourcen-Identifikation (RFID-Technologie) und statische Daten (z.B. Füllstand des Fasses) beinhalten, um oben erwähnte Ziele zu erreichen. Allerdings können die Knoten nur miteinander kommunizieren, wenn Behälter in der Nachbarschaft auch erkannt werden. Jedoch tritt hier das Problem der Nachbarschaftsgröße auf. Zum einen muss man die Größe so klein wie möglich halten, um den Austausch von „proximity-informations“ zwischen den Knoten so gering wie möglich zu halten, was natürlich auch dem energiesparenden Aspekt zu Gute kommt. Zum anderen darf man den Bereich nicht zu klein wählen, da sonst eventuell wichtige Knoten ausgeschlossen werden und damit evtl. Gefahrensituationen nicht erkannt werden. Jeder Knoten berechnet nun mit Hilfe dieser zugesendeten Informationen seine Entscheidung, ob beispielsweise die Lagerplatzkapazität überschritten wurde oder ob beide Fässer nicht nebeneinander gelagert werden dürfen, und sendet seine Entscheidung über sein „wireless-communication-interface“ in das Netzwerk, sowie zum Hauptrechner. Des Weiteren gibt der Knoten seinen internen Zustand über die eingebauten LED-Leuchten bekannt. Diese Aktivitäten werden unter dem Stichwort des lokalen „Monitoring-Prozess“ zusammengefasst. Die Konsistenz der Entscheidungen der einzelnen Knoten hängt allerdings von zwei Faktoren ab. Zum einen von einer zuverlässigen Kommunikation zwischen den Knoten und zum anderen von der Reaktionsgeschwindigkeit der einzelnen Knoten.

4.2 Energiesparende Konzepte im Projekt „Smart Drums“

Um sich nun einzelne Energie verbrauchende Komponenten eines kollaborativen Netzwerkes näher anzuschauen, muss man sich darüber im Klaren sein, das man es hier nicht mit einem statischen Netzwerk zu tun hat. Einmal kann hier je nach Anwendung dynamisch die passende Businesslogik auf die einzelnen Sensoren

herunter geladen werden und zum anderen sind die einzelnen Knoten nicht an einem fixen Standort installiert. Für die Umsetzung eines dynamischen Konzeptes kommt wieder das Service-Konzept zum Einsatz. Das Framework eines jeden Knoten setzt sich aus „Base Services“ und individuellen „Collaborative Services“ zusammen. In den Base-Services sind die Kommunikation (Routing, Datentransport), Synchronisation und Lokalisation wieder zu finden. In der zweiten Gruppe sind die einzelnen Services zur Berechnung des jeweiligen Wirtschaftsprozesses, die evtl. die Grundlage für weitere Berechnungen in anderen Knoten darstellt. Will man nun den Energieverbrauch der einzelnen Knoten untersuchen, stellen diese Klassen und deren Umsetzung einen guten Einstiegspunkt dar. Zwar ist die Umsetzung eines energiesparenden Konzeptes eine grundlegende Anforderung an dynamische Netzwerke, jedoch stehen diesem Konzept weitere Anforderungen gegenüber, die oftmals den Energieverbrauch erhöhen. So ist es für ein effektives Arbeiten zwischen den Sensoren nötig, das nur wirklich relevante Information übertragen wird. Durch geringe Kommunikation wird das Netzwerk nicht zu stark belastet, was letztendlich auch wieder dem energiesparenden Konzept zu Gute kommt. Skalierbarkeit ohne Einbußen in der Netzwerkperformance ist eine weitere Anforderung, welche aber oft mit dem energiesparenden Gedanken kollidiert. Des Weiteren ist man daran interessiert, dass empfangende Knoten eine evtl. fehlerhafte Übertragung erkennen und korrigieren können. Diese Anforderung erhöht zwar die Sicherheit im Netz, jedoch stellt dies einen zusätzlichen Energieaufwand dar. Die aber wohl wichtigste Anforderung ist die universelle Einsatzfähigkeit der Knoten, was sowohl den Verzicht auf Infrastruktur wie Kabel bedeutet, als auch die Situationsbezogene dynamische Speicherung von Quellcode. Der service-orientierte Ansatz wird von zwei unterschiedlichen Plattformen im „Smart-Drums-Projekt“ realisiert (TecO’s Smart-ITS Particles, Ambient Systems’s µNodes). Diese unterschiedlichen Ansätze sollen nun auf ihre Energieeffizienz hin untersucht werden, sowie ihre grundsätzliche Funktionsweise.

4.2.1 Duty-cycling

Grundsätzlich stellen alle zwei Varianten die Möglichkeit des „duty-cycling“ zur Verfügung, mit der man wie oben schon ausführlich beschrieben, den durchschnittlichen Energieverbrauch erheblich senken kann. Allerdings muss die Frequenz des duty-cycle groß genug sein, um eine Veränderung in der Umgebung/Nachbarschaft zu erkennen und angemessen darauf reagieren zu können. Des Weiteren ist hier die Synchronisation der Nachbarknoten ein sehr großes Problem. Auf Grund der Tatsache, dass jeder Knoten unterschiedliche Applikationen abarbeitet, brauchen diese auch unterschiedliche Arbeitszyklen. Somit ist der Ansatz des synchronisierten „duty-cycling“ nicht vorteilhaft. Man müsste sich sonst auf Zyklen einigen, die für brechnungsarme Knoten zu lang sind und diese somit unnötig lange im aktiven Zustand Energie verschwenden.
Ein asynchroner Duty-cycle, bei dem jeder Knoten individuell seinen Schlaf- und Arbeitszyklus wählt, ist hier ebenfalls nicht von Vorteil. Zum einen kann es passieren, dass ein Sensor wach ist und sich ein Fass in seiner Umgebung befindet, aber der Sensor dieses Fasses gerade schläft. Somit kann der aktive Sensor nicht erkennen, um welches Fass es sich handelt und somit auch keine Aussagen darüber machen, ob eine Gefahrensituation (unverträglicher Inhalt, Überschreitung der Lagerkapazität) vorliegt. Selbst wenn beide Sensoren gleichzeitig in einem Zyklus aktiv sind und beide erkennen, dass keine Gefahrensituation vorliegt, genügt es im nächsten Zyklus nicht zu sagen, dass dieses Fass noch das gleiche wie im vorherigen Zyklus ist. Hier könnte ja der Fall eingetreten sein, dass ein Fass durch ein anderes ausgetauscht wurde und sich das neue Fass wieder im Ruhezyklus befindet. Deshalb wäre hier ein duty-cycle von Vorteil, bei dem alle Sensoren eine gemeinsame Aufwachzeit besitzen, aber unterschiedliche Schlafzeitpunkte. Somit beginnt jeder Zyklus zur gleichen Zeit endet aber individuell. Zwar haben nun einige Sensoren weniger schlaf als andere, aber man beugt hier dem Problem vor, dass ein Objekt nicht erkannt wird. Der Wahl der Aufwachzeiten kommt hier eine entscheidende Rolle zu. Werden beispielsweise die Sensoren alle fünf Minuten aufgeweckt, so kann es je nach Dauer der jeweiligen Bearbeitung sein, dass ein Sensor zwar sehr schnell ein neu hinzu gekommenes Fass erkennt, jedoch die Ruhezeit sehr kurz ist. Demnach sind Zyklen von mehreren Stunden für die Erkennung schlecht, aber gut fürs Energie sparen. Somit muss man je nach Situation und Gefahrenpotential eine Obergrenze angeben, in der man auf jeden Fall ein falsch platziertes Objekt oder eine Überschreitung der Lagerkapazität erkennen muss. Diese Obergrenze stellt dann das Intervall der Aufwachzeiten dar. Je größer das Intervall ist, desto mehr Zeit bleibt für den Ruhezyklus. Dieser Duty-cycle gilt aber nur für Fässer, die sich schon im Lager befinden. Neu hinzu gekommene Fässer bleiben so lange im Aktivstatus, bis sie erkannt wurden und die gemeinsame Weckzeit übersendet bekommen haben. Des Weiteren sind die Sensoren durch eine gemeinsame Aufweckzeit in der Lage, gemeinsam zu kommunizieren und somit auch Daten von einem Sensor zum Zentralrechner durch zu schleusen. Dies ist gerade dann wichtig, wenn ein Fass im Lager falsch platziert wurde, sich die Knoten aber gerade im Schlafzyklus befinden. Wenn nun die Knoten aufwachen, wird zwar das falsch platzierte Fass erkannt, jedoch ist nicht sicher gestellt, dass ein Mitarbeiter den Signalton hört. Deshalb muss diese Gefahr an einen Leitrechner gesendet werden, der entsprechende Mitarbeiter verständigt.
Ein ergänzender, wenn auch primitiver Vorschlag, für die Wahl eines Duty-cycle, wäre das komplette Netzwerk für eine gewisse Zeit schlafen zu legen. Diese Möglichkeit kommt aber nur in Betracht, wenn die folgenden zwei Bedingungen erfüllt sind. Zum einen müsste bekannt sein, wann in einem Lager überhaupt ein- und ausgelagert wird, beispielsweise von 8 Uhr bis 19 Uhr. Somit wäre es unter der Bedingung, dass in der restlichen Zeit das Lager nicht betreten werden kann, möglich, das Netzwerk von 19 Uhr bis 8 Uhr schlafen zu legen. Allerdings

nur dann, wenn auch Bedingung zwei erfüllt ist. Diese bezieht sich darauf, das nur auf Unverträglichkeit und Überschreitung der Lagerkapazität geprüft wird und nicht, ob beispielsweise giftige Gase austreten. Die Wahl dieses Duty-cycle ist aus folgenden Gründen zu vertreten. Da das Lager geschlossen ist können nun keine neuen Fässer mehr eingelagert werden und somit können auch keine neuen Gefahrensituationen mehr eintreten. Der Gefahr, dass sich das Netzwerk schlafen legt und eine Gefahrensituation zum Zeitpunkt der Lagerschließung nicht erkennt, wird dadurch vorgebeugt, das alle Sensoren zum Zeitpunkt der Schließung aufgeweckt werden und eine finale Abtastung des Lagers statt findet.
Somit wäre es möglich, zwei Duty-cycles in einem Sensor zu programmieren. Den ersten vorgestellten Duty-cycle für die Zeit, wenn das Lager geöffnet ist und zum anderen den Duty-cycle, wenn das Lager geschlossen ist.

4.2.2 zusätzliche Energiesparmechanismen

Alle Plattformen verwenden handelsübliche Batterien der Größe AA oder AAA. Der Einsatz von wideraufladbaren Batterien wird nicht in Betracht gezogen, da sie auf Grund der Größe oder der Anschaffungskosten nicht rentable genug sind. Außerdem bräuchte man eine Ladestation, in der die Batterien periodisch aufgeladen werden, was die universelle Einsatzfähigkeit der Smart-Drums erheblich einschränken würde. Problematisch ist bei handelsüblichen Batterien, dass ihre Leistungsfähigkeit mit der Temperatur schwankt. Zu niedrige oder zu hohe Temperaturen beeinflussen das Verhalten negativ. Der Einsatz speziell entwickelter Batterien für dieses Szenario währe eine mögliche Lösung, um eine geforderte Lebenszeit von fünf bis sechs Jahren zu erreichen. Dies wäre jedoch mit immensen Entwicklungs- und Anschaffungskosten verbunden, welche groß angelegte Überwachungsszenarien zu teuer machen würden. Ein weiterer Ansatz um sich temperaturspezifischen Einwirkungen entgegen zu stellen, wäre die Verwendung eines isolierenden Gehäuses, welches gerade im Außenbereich in der Lage ist, Wärme auf zu nehmen und diese bei sinkenden Temperaturen wieder ab zu geben. Bei Lagerung im Innenbereich dürfte eine konstante Temperatur über eine Heizung möglich sein. Dies ist aber nur dann relevant, wenn die zu lagernden Stoffe diese Temperatur auch vertragen. Kurz gesagt, was nützt eine optimale Batterietemperatur, wenn die Stoffe dadurch explodieren und es somit keinen Grund der Überwachung mehr gibt. Temperaturschwankungen beeinflussen aber nicht nur die Batterie, sonder auch die Hardware. Hier kann eine nicht-optimale Temperatur einen erhöhten Energieverbrauch zur Folge haben, was dann die Batterie zusätzlich belastet.

4.2.2.1 TecO's Smart-IST Particles

TecO's Smart-ITS Particles stellt eine flexible Plattform für hochdynamische Anforderungen zur Verfügung. Die Flexibilität wird durch eine Trennung von Kommunikations- und Sensoreinheit erreicht, womit ein einfaches Austauschen der einzelnen Einheiten möglich ist, um die Plattform auf den aktuellen Einsatzort anzupassen. Die Kommunikation wurde für eine mobile ad-hoc Kommunikation konzipiert, um eine Kommunikation ohne Infrastruktur zu ermöglichen. Es steht eine enorme Rechenpower für die Bearbeitung gemeinschaftlicher Tasks (Kollaboration Framework) zur Verfügung.
Als zusätzlichen Energiesparmechanismus wird hier im Rahmen der „AwareCon" das Konzept des „early-shutdown" angeboten. Dies ermöglicht es den empfangenden Knoten, eine Datenübertragung frühzeitig abzubrechen, wenn die zu empfangenden Daten unwichtig sind. Somit kann der sendende und der empfangende Knoten seine Sendeeinheit abstellen oder sich, wenn möglich, komplett schlafen legen, bis sein nächster „Duty-cycle" beginnt. Durch die Trennung von Kommunikations- und Sendeeinheit, ist es möglich, dass der Knoten Sensordaten verarbeiten kann und seine Kommunikationseinheit in den Zustand „inaktiv" versetzen kann.
Hier wäre es nun mittels eines zweiten duty-cycles möglich, in einem individuell gewählten Arbeitszyklus zu schauen, ob Daten zu empfangen sind. Dies würde eine weitere Möglichkeit zur Verfügung stellen, um die energieintensive Kommunikation zu senken und somit kostbare Energie zu sparen. Jedoch hängt die maximale Lebenszeit bei einem service-orientierten Konzept maßgeblich von der Applikation ab, die auf dem Sensor ausgeführt wird und ob diese eine erhöhte Kommunikation voraus setzt.

4.2.2.2 Ambient Systems's µNodes

Im Gegensatz dazu stellt Ambient Systems's µNodes einen energiesparenden Zwei-Prozessoransatz vor. Ein Prozessor kümmert sich um die Kommunikation, der andere übernimmt die Stellung des Applikations-Prozessors ein. Auf dem Applikationsprozessor läuft das echtzeitfähige Betriebssystem „AmbientRT", welches eine Ausgewogenheit zwischen der Taskausführung und energiesparenden Schlafzuständen sicherstellt. Durch die Bearbeitung in Echtzeit ergeben sich neue Möglichkeiten, in der realitätsnahen Umsetzung der Prozesslogik und deren Integration in vorhandene Back-End-Systeme.
Um Energie zu sparen wird zum einen beiden Prozessoren die Möglichkeit gegeben, in einen „ultra-low-power-Modus" zu wechseln, wenn sie nichts zu erledigen haben. Im Bereich der Kommunikation wird zudem ein optimiertes Routingprotokoll verwendet, das eine Versendung von Daten über mehrere unterschiedliche Knoten ermöglicht und den Energieverbrauch auf ein Minimum zu reduzieren. Dieser Mechanismus basiert auf einer „Multi-hop ad-hoc Netwerkkommunikation".

Allerdings müssen auch hier Prozesse für die Lokalisierung im Hintergrund ablaufen, um zu erkennen ob neue Knoten hinzugekommen sind. Diese Plattform stellt eine ideale Lösung für Sensornetzwerke dar, da diese klein, günstig und sehr energieeffizient sind und darüber hinaus noch sehr leistungsfähig in der Berechnung, Kommunikation und Sensorik. Dennoch birgt diese Variante einen enormen Nachteil im Gegensatz zur ersten Plattform. Hier wird ein Zwei-Prozessor-System verwendet, was unter dem Energiespargesichtspunkt nicht ideal ist. Der Einsatz von mehreren Prozessoren rentiert sich bei sehr viel Berechnungs- und Kommunikationsaufwand, jedoch schon alleine das Hochfahren beider Prozessoren aus dem Passivmodus stellt einen signifikanten zusätzlichen Energieaufwand dar. Außerdem ist die zur Verfügung stehende Anzahl von Sensoren beim TecO's Smart-ITS Particle viel größer, als bei der zweiten Plattform, was die Wahrscheinlichkeit für eine realitätsnahe Abbildung der Umwelt erhöht. Zwar mag die zweite Plattform sehr energie-effizient sein, liefert aber schlechtere Ergebnisse in der Kategorie Mobilität und Flexibilität. Gerade das Multi-Hop-Routing setzt voraus, das mehrere Knoten aktiv sein müssen, um die Information über mehrere Knoten zu versenden. Jedoch bringt dieses energiesparende Routing nichts, wenn in der Nachbarschaft nur wenige Knoten aktiv sind.

Der Einsatz beider Plattformen kann nicht im Voraus entschieden werden, sondern muss nach Gesichtspunkten der aktuellen Situation getroffen werden. In Abb. 21 und 22 sind beide Plattformen und ihre Vorzüge zu sehen, außerdem wird noch eine dritte Plattform von Sinidron mit aufgeführt, die aber im Bereich der „Smart-Drums" nicht eingesetzt wird und von daher nicht diskutiert wird.

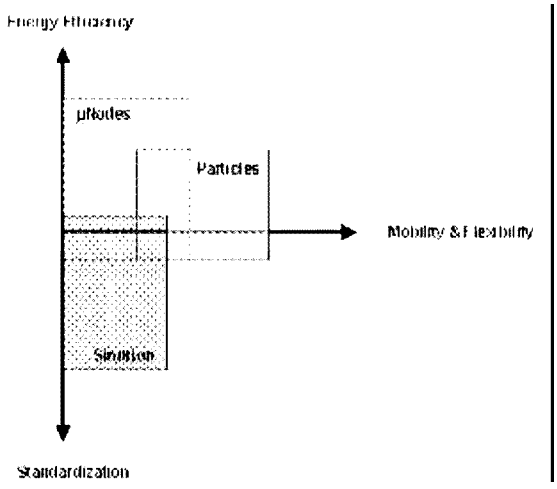


Abb. 21 Stärken/Schwächen der Plattformen

Platform	µNode v1.0 / v1.1	Particle / SPart	Sindrion
Program Memory (KB)	60 / 48	128 / 32	2048
RAM (KB)	2 / 10	4 / 1.5	128
External non volatile memory (KB)	No / 512	512 / 32	No
Integrated Sensors	Temperature, light / Temperature	No / Temperature, light, microphone, 3D acceleration, force	No
Actuators	LED(4), LCD	LED(2), speaker / LED(2)	LED
Radio data rate (kbps)	76.4	125	50
Physical dimensions (mm^2)	51x32 / 40x50	15x48 / 18x37	55x66

Abb. 22 Technische Details der unterschiedlichen Plattformen

5 Referenzen

[1] INOBAT Interessenorganisation Batterieentsorgung, "Wie funktioniert eine Batterie?", http://www.inobat.ch/fileadmin/user_upload/pdf/Wie_d.pdf

[2] Dirk Uwe Sauer, „Entladung-Temperatur-, Strom- und Alterseinflüsse", Fraunhofer-Institut für Solare Energiesysteme ISE, http://www.ise.fraunhofer.de/german/fields/field3/mb3/publications/beitrag_010416.pdf

[3] S. Park, A. Savvides, M. B. Srivastava, "Battery Capacity Measurement and Analysis using Lithium Coin Cell Battery", International Symposium on Low Power Electronics and Design, pp. 382-387, Huntington Beach (California) 2001

[4] T.F. Fuller, M. Doyle, J. Newman, "Simulation and Optimization of the Dual Lithium Ion Insertion Cell", Journal of Electrochem. Soc., vol. 141, no. 4, Apr. 1994, pp

[5] H.D. Linden, Handbook of Batteries, 2nd ed., McGrawHill, New York 1995

[6] M. Pedram, Q. Wu, "Battery-Powered Digital CMOS Design", Proceedings of Design, Automation and Test in Europe Conference and Exhibition, Munich, Germany, March 1999

[7] C. F. Chiasserini and R. R. Rao, “Pulsed battery discharge in communication devices”, Proceedings of Mobicom 99, Seattle, August 1999
[8] Q. Wu, Q. Qiu, M. Pedram, “An Interleaved Dual-Battery Power Supply for Battery-Operated Electronics”, ASP-DAC-00: IEEE Asia and South Pacific Design Automation Conference, Yokohama, Japan, pp. 387-390, Jan. 2000
[9] L. Benini, A. Macii, M. Poncino, “Discharge Current Steering for Battery Lifetime Optimization”, ISLPED´02, August 12-14, 2002, Monterey, CA, USA
[10] L. Benini, G. Castelli, A. Macii, R. Scarsi, “Battery-Driven Dynamic Power Management”, IEEE Design & Test of Computers, Vol. 18, No. 2, pp. 53-60, March-April 2001
[11] K. Lahiri, A. Raghunathan, S. Dey, D. Panighrahi, “Battery-Driven System Design: A new Frontier in Low Power Design”, Proc. ASP-DAC/Int. Conf. VLSI Design, pp. 261-267, Jan. 2002
[12] T. Li, L. K. John, “Run-time Modelling and Estimation of Operating System Power Consumption”, Proceedings of the International Conference on Measurement and Modeling of Computer Systems, 2003
[13] J. Liu, P. H. Chou, “Power-Aware Scheduling under Timing Constraints for Mission-Critical Embedded Systems”, Second International Workshop, PACS 2002 Cambridge, MA, USA, pp. 84-98, 2002
[14] D. Rakhmatov, S. Vrudhula, C. Chakrabarti, “Battery-Conscious Task Sequencing for Portable Devices Including Voltage/Clock Scaling”, *DAC 2002*, June 10-14, 2002 New Orleans, Louisiana, USA
[15] S. Ren, Q. Li, H. N. Wang, X. Chen and X. Zhang, „Probabilistic Coverage for Object Tracking in Sensor Networks”, in Mobicom 2004 Poster Session, 2004
[16] C. Gui and P. Mohapatra, “Power Conservation and Qualitiy of Surveillance in Target Tracking Sensor Networks”, ACM Mobicom, 2004
[17] Q. Cao, T. Abdelzaher, T. He, J. Stankovic, “Towards Sleep Scheduling in Sensor Networks for Rare-Event Detection”, International Conference on Information Processing in Sensor Networks (IPSN'05), April 2005.
[18] R. Subramanian, F. Fekri, “Sleep Scheduling and Lifetime Maximization in Sensor Networks: Fundamental Limits and Optimal Solutions”, IPSN06
[19] W. Ye, J. Heidemann and D. Estrin, An Energy-Efficient MAC Protocol for Wireless Sensor Networks”, in INFOCOM 2002
[20] T. van Dam, K. Langendoen, “An adaptive energy-efficient MAC Protocol for Wireless Sensor Networks”, ACM Press, 2003
[21] A. Giridhar, P.R. Kumar, “Maximizing the Functional Lifetime of Sensor Networks”, Proceedings of the 4th international symposium on Information processing in sensor networks, L.A., California, 2005