Wireless Sensor Networks

Seminararbeit

von

Alexander Holzinger

vorgelegt am

Lehrstuhl für Praktische Informatik IV

Prof. Dr. W. Effelsberg

Universität Mannheim

Februar 2002

Betreuer : Dr. Hartmut Ritter, Universität Karlsruhe

Inhaltsverzeichnis

Abbildungsverzeichnis

Wireless Sensor Networks

Alexander Holzinger

Kurzfassung

Diese Seminararbeit befasst sich mit der Technologie der Wireless Sensor Networks, indem sie einen Überblick über vorhandene und visionäre Techniken gibt.

Zuerst wird ein Überblick über die verwendete Technik, Aufgaben und Probleme gegeben. Danach folgen mögliche Lösungsansätze dieser Probleme und zum Abschluss wird ein noch in der Planung stehendes Anwendungsbeispiel vorgestellt.

1 Einleitung

1.1 Definition, Technik

Unter einem Wireless Sensor Network versteht man den Zusammenschluss von mehreren, kleinen Batterie-betriebenen Sensoren, die Messdaten und Informationen sammeln, verarbeiten und dem Benutzer zur Verfügung stellen. Die Sensoren sind dabei topologisch meist regional und hierarchisch geordnet. Dabei stehen jeweils „Cluster Heads“ an der Spitze, die den eigentlichen Backbone bilden. Die einzelnen Sensoren auf unterster Ebene kommunizieren meist nur untereinander bzw. mit dem Cluster Head über kurze Funkstrecken (z.B. Bluetooth [BLUE]). Bei den Cluster Heads dagegen kommt normalerweise der IEEE 802.11 Standard zum Einsatz (2.4 GHz Frquenzband, 11 Mbps Datenrate) :

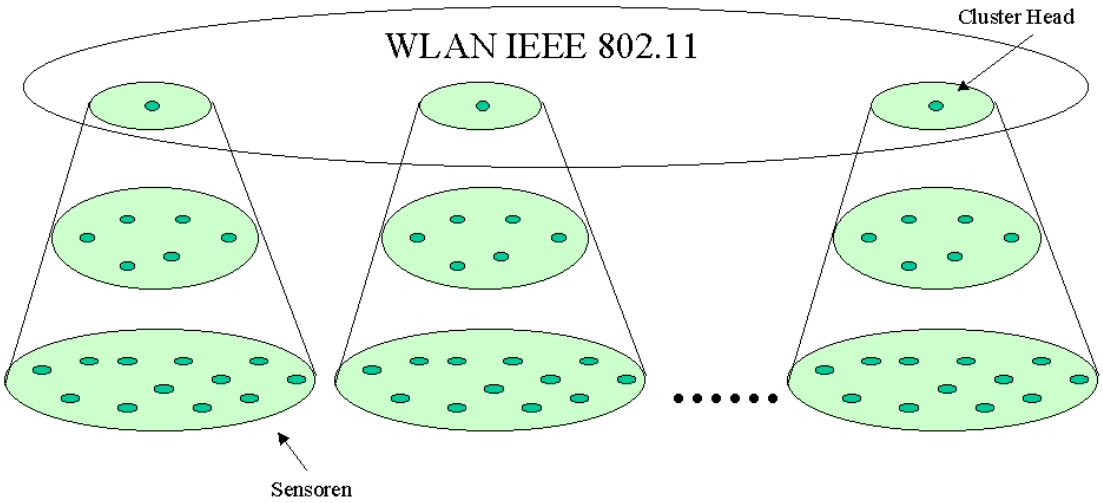


Abbildung 1 : Typisches Wireless Sensor Network

Die einzelnen Sensorknoten haben dabei beispielsweise folgenden Aufbau (Bild 2) :

- Batterie als Stromversorgung, incl. eines DC-DC Spannungswandlers
- Analoge Messdaten werden digitalisiert und von einem Prozessor verarbeitet
- Daten werden über einen Funk-Transceiver versendet bzw. empfangen
- Ein Betriebssystem incl. Sensor Algorithmen und Netzwerkprotokollen ist resident im ROM

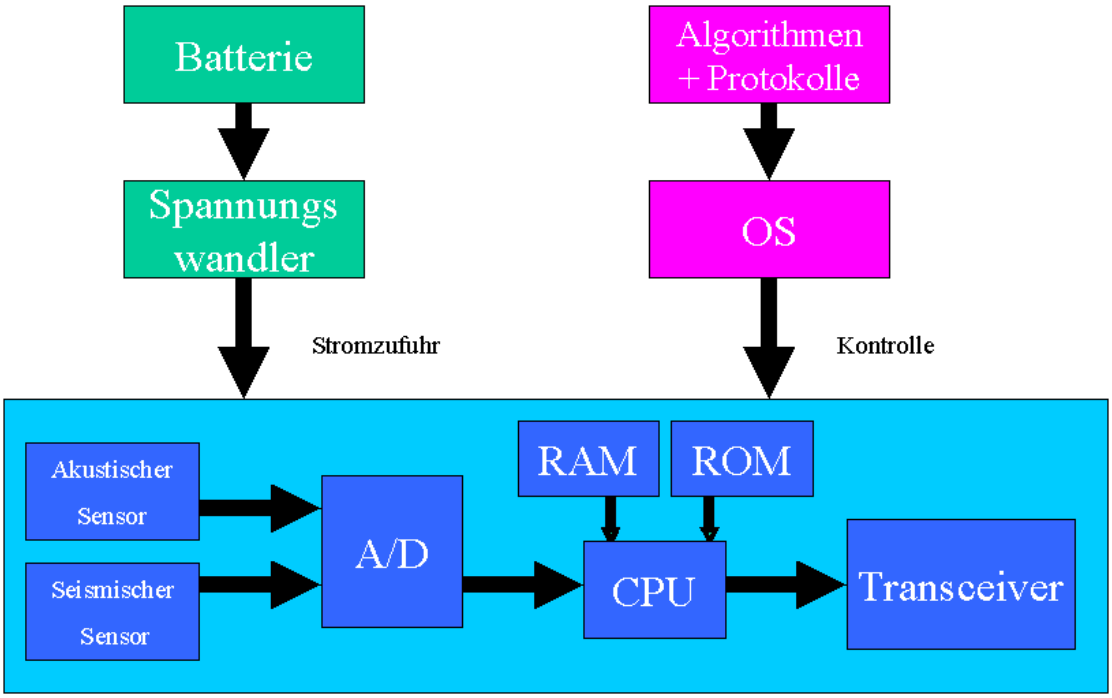


Abbildung 2 : Sensorknoten Architektur [MBCS+01]

1.2 Aufgaben, Nutzen

Die Aufgabe der Sensoren besteht natürlich im Sammeln von Informationen jeglicher Art, bspweise Audio, Video oder Umweltmessdaten und der Verarbeitung bzw. dem Versenden dieser Daten an andere Knoten bzw. den Endnutzer. Das intelligente Verarbeiten und Datenmanagement dieser Informationen durch das Netzwerk ermöglicht es dem Benutzer nicht nur auf einzelne Messdaten oder Audio- und Videoströme zuzugreifen, sondern auch langfristige Strategien zu verfolgen, wie bspweise das Auswerten der Daten mittels Datenbankanfragen und Datamining oder sogar das Verfolgen des Lernprozesses von Kindern in einem Kindergarten (siehe Kap 7).

Der physikalische Einsatzort der Sensoren beschränkt sich dabei nicht nur auf Räume und Gebäude, sondern auch auf unwirtliche Regionen, die schwer zugänglich sind. Somit gibt es sowohl bedingt durch die Aufgaben der Sensoren, als auch durch deren Einsatzort eine Reihe von Problemen, die im folgenden geschildert werden sollen.

1.3 Probleme

Hier soll nur eine kurze Erwähnung und Einordnung der Probleme im Umgang mit den Sensoren erfolgen, eine ausführlichere Behandlung und Lösungsansätze erfolgen in den jeweiligen Kapiteln.

Probleme bedingt durch die Netzwerkarchitektur :

- das Implosion- und das Overlap-Problem. Lösungsansatz durch das Middleware-Protokoll SPIN und die Middleware-Architektur SINA (siehe Kap. 2 bzw. 3)

Probleme bedingt durch die Hardware und das Einsatzgebiet :

- die beschränkte Energie der Sensoren bedingt durch die Stromversorgung über Batterien. Lösungsansatz durch Dynamic Voltage Scaling, Radio Communication Hardware und Energy-Efficient Networks (Kap. 4)

2 Das Sensor Protocol for Information via Negotiation (SPIN)

Das folgende Kapitel beschreibt das Sensor Protocol for Information via Negotiation [HKBa99], das 1999 am MIT als Middleware Protokoll (also unterhalb von Anwendungen durch Bibliotheken zugänglich) entwickelt wurde und dazu dienen soll, das Implosion und Overlap-Problem in energie-beschränkten Wireless Sensor Netzwerken zu verhindern. Zunächst werden dazu die erwähnten Probleme geschildert.

2.1 Das Implosion-Problem

Sowohl in herkömmlichen verdrahteten Netzwerken, wie auch in den Wireless Sensor Networks, gab es eine Reihe von Protokollen die beim Senden von Daten an andere Empfänger ein Problem, Flooding genannt, verursachen. Ausgangspunkt dabei ist das Senden von Daten eines Knotens an alle Nachbarn. Die Empfänger senden nun ihrerseits wieder eine Kopie dieser Daten an alle ihrer Nachbarn. So kommt es im Netz zu einer regelrechten Überschwemmung mit ein und derselben Nachricht. Als Folge davon entsteht das bereits erwähnte Implosion-Problem. Dabei ist in den Knoten, als Folge des Flooding, eine Nachricht mehrfach vorhanden. Dadurch werden sowohl knoteninterne Ressourcen, als auch Bandbreite (beim Versenden der doppelten Daten) verschwendet. Dies wird auch nochmals in Abbildung 3 auf der nächsten Seite verdeutlicht :

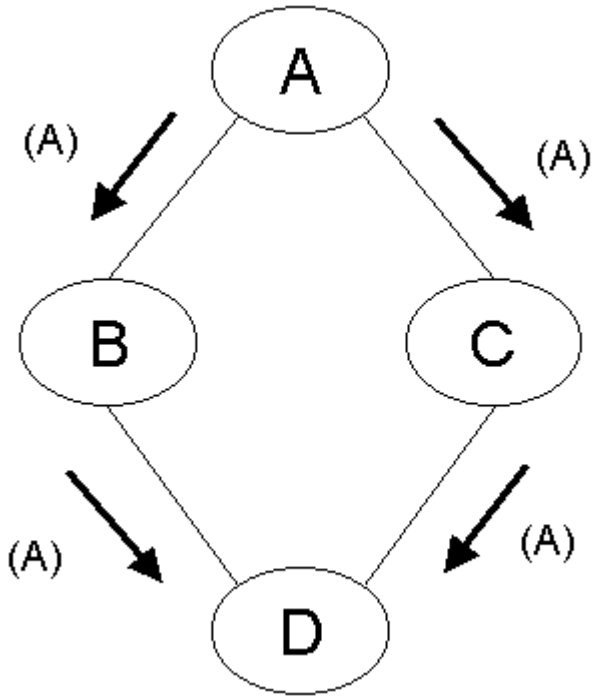


Abbildung 3 : Das Implosion-Problem: Alle Knoten senden die Daten an alle Nachbarn, so dass bei D zwei Kopien der Daten von A eintreffen. Diese Kopien belasten Knoten D im Bezug auf interne Ressourcen und Bandbreite beim Versenden der Daten von D an andere Knoten

2.2 Das Overlap-Problem

Die Beobachtungsgebiete der Sensorknoten, in denen sie Messdaten sammeln, sind meist überlappend angeordnet, so dass das Versenden dieser Messdaten an alle Nachbarn wieder zu selbigen Problem wie oben führt. Das Overlap-Problem ist noch schwerer zu bewältigen als Implosion, da hier nicht nur die Netzwerktopologie für das Problem verantwortlich ist, sondern auch die Zuordnung der gewonnenen Daten zu den Knoten zu beachten ist (Abb. 4 s.u.).

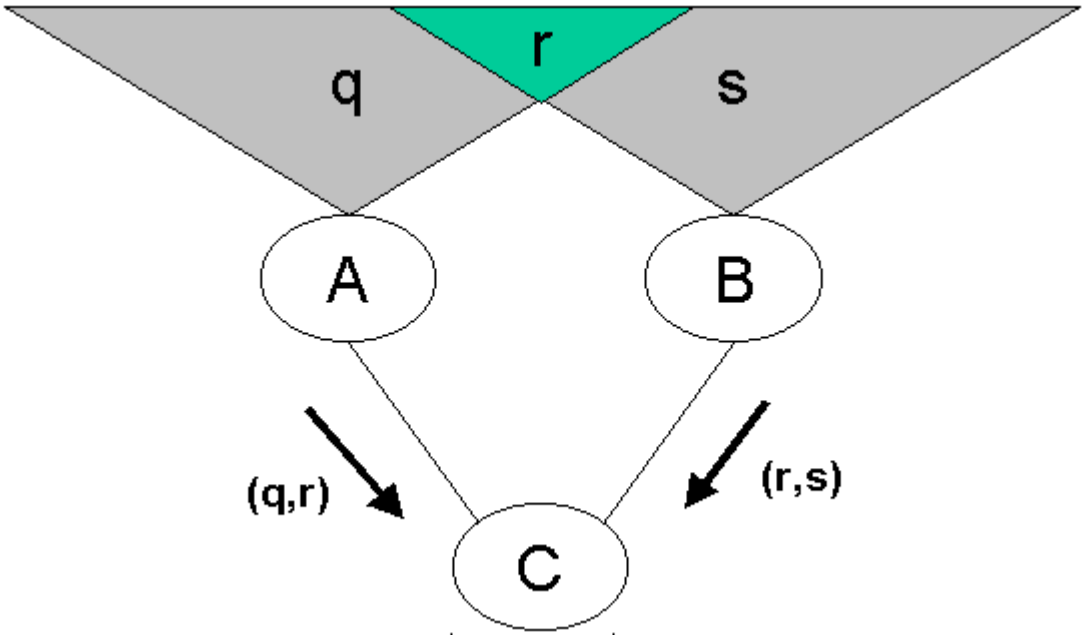


Abbildung 4 : Das Overlap-Problem :Sowohl Knoten A als auch B senden Infos über Region r, da sich die Einzugsgebiete der beiden Knoten dort überlappen. Diese Daten landen dann doppelt bei Knoten C

2.3 Das SPIN-Protokoll

Den genannten Problemen tritt man hauptsächlich mit der Hilfe von Kommunikation zwischen den Knoten vor dem Versenden der eigentlichen Daten und dem Verwenden von Meta Daten entgegen. Desweiteren besteht auch noch die Möglichkeit Daten energiesparend zu verarbeiten bzw. zu verschicken, worauf hier aber nicht eingegangen werden soll.

2.3.1 Meta Daten

Vor dem Versenden der Nutzdaten kommunizieren die benachbarten Knoten des Netzwerks miteinander, um sich darüber klar zu sein, ob ein Knoten die Daten die ein anderer verschickt überhaupt braucht bzw. schon hat. Somit braucht man eine Konstruktion um die Nutzdaten einheitlich beschreiben und benennen zu können. Dazu verwendet man die Meta Daten.

SPIN schreibt dabei kein einheitliches Format für die Meta Daten vor, sondern überlässt dies der jeweiligen Anwendung die oberhalb von SPIN läuft. Beispielsweise könnte eine Meta-Data x eines Messdatensensors bedeuten, dass es sich um Nutzdaten des Messdatensensors x handelt. Dagegen könnte (x,y,z) bei einem Kamerasensor bedeuten, dass es sich um Nutzdaten der Region (x,y) mit Orientierung z handelt.

Die Datengrösse in Bytes der Meta Data x für die Daten X muss natürlich kleiner sein als die Grösse von X. Die Kosten für die zusätzliche Speicherung der Meta Daten und Kommunikation zwischen den Knoten wirkt aber kaum als Belastung verglichen mit der u.U. überflüssigen Übertragung von grossen Mengen an Videodaten an einen Nachbarknoten der diese bereits erhalten hat.

2.3.2 Kommunikation, SPIN Messages

Es gibt in SPIN drei Typen von Nachrichten zwischen den Knoten :

- ADV : new data advertisement : wenn ein Knoten neue Nutzdaten X zum Versenden hat, kann er dies durch Verschicken der ADV (x) Nachricht an alle Nachbarknoten kundtun.
- REQ : request for data : Ein Knoten verschickt diese Nachricht, falls er neue Daten haben möchte
- DATA : data message : enthält aktuelle Nutzdaten

2.3.3 Das eigentliche Protokoll

Das SPIN Protokoll ist ein einfaches 3-Stufen Handshake Protokoll, bei dem jede Stufe einer Nachricht wie oben beschrieben entspricht (siehe auch Abb.5) .

Das Protokoll startet damit, dass ein Knoten neue Nutzdaten zur Verfügung hat und diese an alle Nachbarn schicken möchte. Dies zeigt er durch das Versenden einer ADV Message an, in der die Nutzdaten durch Meta Daten beschrieben werden. Die Nachbarknoten untersuchen nun ob sie die Daten schon angefordert oder empfangen haben. Falls nicht, schicken sie eine REQ Message an den Ursprungsknoten zurück. Schließlich schickt dann der Ursprungsknoten jeweils per Unicast die Nutzdaten an all diese Nachbarn (die sich mit der REQ Message bei ihm gemeldet haben) mit Hilfe einer DATA Message.

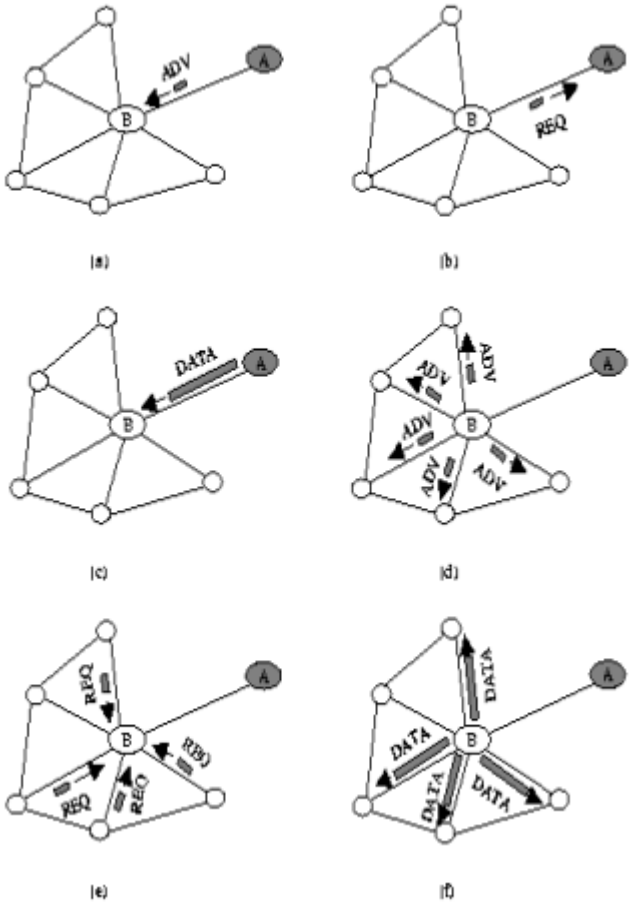


Abbildung 5 : Knoten A startet mit einer ADV Message an Nachbar B (a). Dieser antwortet mit einer REQ an A (b). A schickt B die Daten (c). Dieser führt dies fort mit seinen Nachbarn (d). Diese ordern die Daten ihrerseits (e,f). Einer der Knoten tut dies nicht, da er die Daten schon vorher zur Verfügung hatte. [HKBa99]

In obiger Abbildung sieht man auch, dass Knoten B mit Hilfe einer Aggregation der Daten die er von A geschickt bekommt und eigenen vorhandenen Daten, eine neue ADV dieser aggregierten Daten zu seinen Nachbarn schicken könnte, anstatt der Orginaldaten von A. Dies ist auch der Fall, falls B die Daten von A schon hat (ADV werden dann trotzdem an alle anderen verschickt).

Der Vorteil des SPIN Protokolls liegt hauptsächlich in seiner Einfachheit. Jeder Knoten muss nur wenig Aufwand betreiben um zu entscheiden, ob er neue Daten braucht oder nicht. Somit wird wenig Rechenleistung benötigt und auch wenig (physikalische) Energie verschwendet. Ausserdem muss jeder Knoten nur seine unmittelbaren Nachbarschaftsknoten und nicht die gesamte Topologie des Netzes kennen. Daraus folgt, dass SPIN in völlig unkonfigurierten Netzen mit kleinen „Startup Costs“ für die „Nearest Neighbor“ Suche und auch in Netzen die häufig die Topologie ändern sehr gut anwendbar ist (daher auch in WLANs).

Es ist offensichtlich, dass das SPIN-Protokoll sowohl das Implosion-, als auch das Overlap-Problem löst. Das Implosion-Problem wird vermieden, indem nur Daten an Knoten weitergegeben werden die daran interessiert sind. Das Overlap-Problem z.B. aus Abb. 4 wird durch trennen der Daten (q,r) bzw (r,s) in ADV(q), ADV(r) und ADV(s) gelöst, so dass Knoten C REQ(q) an A, REQ(s) an B und nur einmal REQ(r) an A oder B schickt.

SPIN kann somit als eine Art „application-level-multicasting“ des traditionellen IP-Multicasting gesehen werden, da auf Anwendungsebene sowohl Topologie, als auch Daten-Layout bekannt, und mit Multicast-Trees vergleichbar sind.

2.4 Testergebnisse

Im Rahmen der Entwicklung des SPIN am MIT testete man dieses Protokoll u.a. gegen das oben beschriebene Flooding und ein „perfektes“ Protokoll, das komplette Kenntnis der Topologie des Netzes und der Anforderungen der Knoten an Datenpaketen hat (Abb.6) :

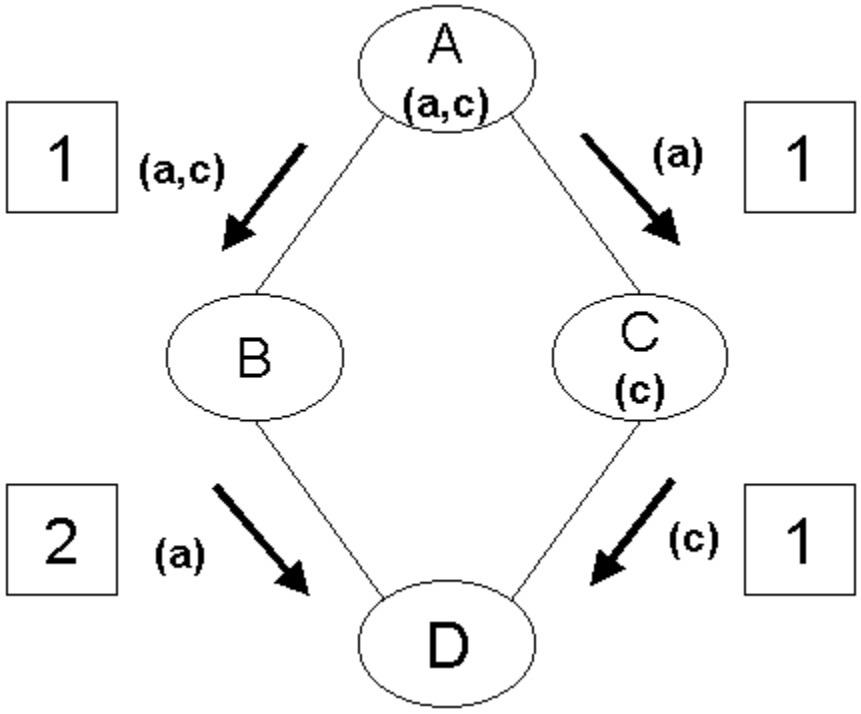
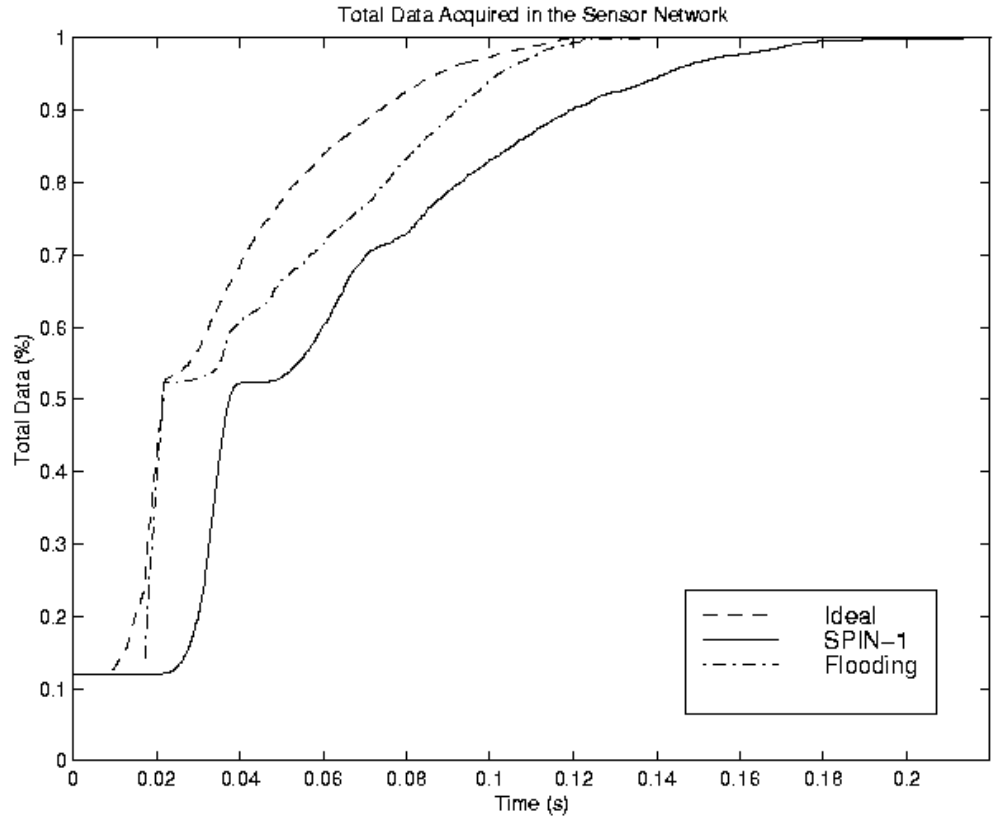


Abbildung 6 : Ideales Protokoll : A möchte Daten a und c versenden, B bekommt beides, C bekommt nur a, da er c schon vorher hatte. D fordert zuerst von c von C und dann a von B. So wird sowohl Implosion (B und C schicken a,c jeweils an D), als auch Overlap (A und C haben beide Info c und versenden diese) verhindert. [HKBa99]

Anhand folgender Graphen sieht man wie SPIN zwar im Vergleich zu Flooding länger braucht um alle Daten im Testnetz zu verteilen („Negotiation“), aber auch wie deutlich weniger Ressourcen verschwendet werden. Ausserdem ist SPIN in der Performance nahe an Ideal.



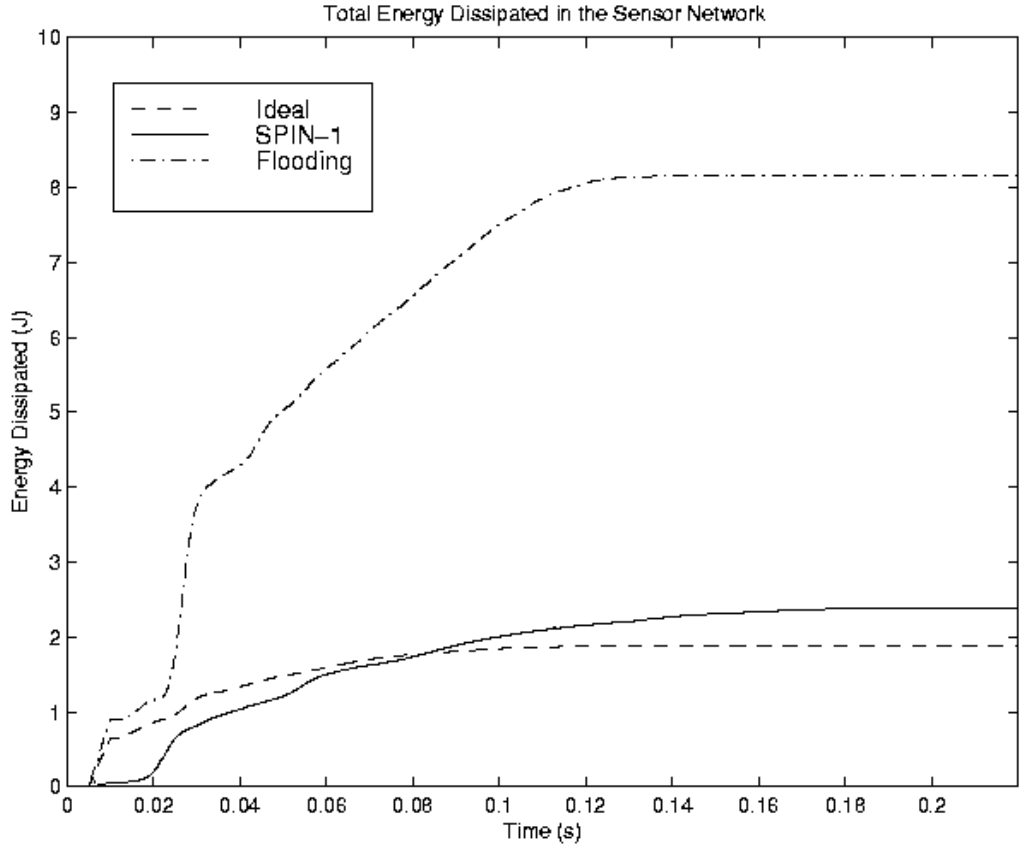


Abbildung 7 und 8 : Performance von SPIN vs Ideal- und Flooding-Protokoll beim Zeitverbrauch für den Empfang der Daten (Abb.7) und Ressourcenverbrauch (Abb.8) [HKBa99]

2.5 Fazit

Zusammenfassend kann damit folgendes festgestellt werden :

- SPIN schneidet gegenüber Flooding in Ressourcen- und Bandbreitenbedarf deutlich besser ab
- SPIN kommt nahe ans Ideal-Protokoll heran
- SPIN vermeidet Overlapping
- SPIN vermeidet Implosion
- SPIN spart Energie, da keine unnötig (grossen) Datenmengen verschickt werden
- SPIN ist somit gut geeignet für Wireless Sensor Networks

3 Das Sensor Information Networking Architecture (SINA)

In den folgenden Abschnitten soll näher auf SINA [SSJa00] eingegangen werden, eine an der University of Delaware entwickelte Architektur für Wireless Sensor Networks, die Protokolle und Dienste der Middleware zur Verfügung stellt.

3.1 SINA Architektur

Einen Schritt weiter als das oben dargestellte SPIN Protokoll geht die Sensor Information Networking Architecture (SINA), da es nicht nur SPIN selbst implementiert, sondern eine Reihe anderer Middleware Services zur Verfügung stellt. Dazu gehört das Verarbeiten von Anfragen, Kommandos, Antworten und Veränderungen des Sensor Netzwerks.

Die der SINA zugrundeliegende Architektur lässt sich in folgende Bereiche gliedern :

- Hierarchical Clustering der Sensorknoten (3.2)
- Attribute-Based Naming (3.3)
- Location Awareness (3.4)

In den nun folgenden Abschnitten sollen diese Dienste näher beschrieben werden

3.2 Hierarchical Clustering

In Abhängigkeit von Nachbarschaft und Energiereserven der Sensorknoten existieren in SINA sowohl Blattknoten auf unterster Ebene und Knoten in den mittleren Ebenen, als auch Cluster Heads auf obersten Ebene. Die Festlegung welcher Knoten welcher Hierarchiestufe angehört kann dabei dynamisch erfolgen, so dass beispielsweise ein Cluster Head mit niedrigem Energiestand durch einen anderen Knoten als Cluster Head ersetzt wird. Vorteil des Clusterns ist, dass eine Aufgabenverteilung stattfindet, die Energie spart, da nicht alle Knoten alle Aufgaben übernehmen müssen.

3.3 Attribute-based Naming

Da die Anzahl der Knoten in einem Sensornetzwerk meist sehr gross ist, sind Anfragen des Benutzers an einzelne Knoten nur schwer zu bewerkstelligen. So könnte ein Benutzer an der Temperatur in der Süd-West-Region interessiert sein oder die Durchschnittstemperatur aller Knoten erfragen wollen, anstatt die Temperatur von Knoten Nr. 102 abzufragen. Um dies zu ermöglichen führt SINA einen attribut-basierten Namensdienst ein.

Beispielsweise beschreibt [type = temperature, location = S-W, temperature = 23] alle Temperatursensoren in der SW-Region mit einer Temperatur von 23 Grad Celsius. Diese Sensoren würden dann auf eine Anfrage der Art „welche Regionen haben eine Temperatur von mehr als 20 Grad“ reagieren.

3.4 Location Awareness

In den grossen Sensornetzwerken ist das Wissen über den Aufenthaltsort eines jeden Knotens meist unabdingbar. Daher gibt es verschiedene Methoden wie dieser Aufenthaltsort ermittelt werden kann.

Zum einen kann über GPS die absolute Position bestimmt werden. Die GPS-Receiver werden aber aus Kostengründen nur in vereinzelte Knoten eingebaut und dann können die anderen Knoten mittels eines Beacon-Signals ihre eigene Position im Bezug auf den „GPS-Knoten“ ermitteln. Weiterhin sind auch optische Verfahren im Einsatz, die zwar sehr genau sind, aber nur in kleinen Regionen anwendbar sind.

Insgesamt können somit mit diesen 3 Diensten sehr viel effizientere Anfragen an das Netzwerk gestellt werden, da alle Daten periodisch und regionenweise gesammelt, verarbeitet und in den Cluster Heads zum schnellen Abruf zur Verfügung gestellt werden können, was wiederum per Namensdienst sehr einfach möglich ist.

3.5 Sensor Query and Tasking Language (SQTL) und das SSE

Als Schnittstelle zwischen den Applikationen und der SINA-Middleware, fungiert die Anfragesprache SQTL, die nicht nur anwendungsbezogene Datenanfragen stellen kann, sondern auch Anfragen bezüglich Hardware (bsp. : getTemperature, turnOn), Ort (bsp. : isNeighbor, getPosition) oder Kommunikation (bsp. : tell, execute) zur Verfügung stellt.

Anfragen und Befehle, die von den Applikationen mittels SQTL an die SINA-Middleware gestellt werden, verarbeitet ein Sensor Execution Environment (SSE). Dieses Modul läuft auf jedem Knoten und nimmt die Zuweisung von Sensor-Namen zu Hardware-Addressen und abstrakten Daten-Namen zu konkret vorliegenden (Mess-)Daten der Hardware vor. Damit entscheidet das SSE auf abstrakter Ebene (noch vor der Routing-Schicht) welche Daten an welche Knoten im Netzwerk weiterzuleiten sind.

Das SSE wendet dabei verschiedene Techniken zur Informationsverarbeitung an, die in den folgenden Abschnitten beschrieben werden.

3.6 Information Gathering Methods

Wie oben erwähnt, entscheidet das SSE welche Daten an welche Knoten weiterzuleiten sind, und versucht somit auch, die in der Einführung erwähnten Probleme des Implosion und der Energie-Knappheit, zu vermeiden.

Dazu werden nun 4 verschiedene Techniken betrachtet :

3.6.1 Sampling Operation (auch APR genannt)

Hierbei entscheidet jeder Knoten anhand einer Wahrscheinlichkeit p, ob er Daten sendet oder nicht. Dabei wird p von den Cluster Heads vorgegeben, in Abhängigkeit der Anzahl der Antworten die aus einem Cluster benötigt werden.

3.6.2 Self Orchestrated Operation

Hier sendet jeder Knoten, allerdings je mit einer Verzögerung d. Dabei ist d hauptsächlich abhängig von der Anzahl der Hops vom Front-End zum jeweiligen Knoten.

3.6.3 Diffused Computation Operation

Hier ist in jedem Knoten eine Aggregations-Logik vorhanden, die dem Knoten sagt ob Daten Richtung Front-End zu versenden sind oder nicht bzw. wie sie zu aggregieren sind um Bandbreite zu sparen. Ein Nachteil dabei ist allerdings die erhöhte Antwortzeit des Knotens, bedingt durch die Verarbeitungszeit der Aggregation.

3.6.4 SPIN

SPIN wurde bereits in Abschnitt 2 diskutiert und kann ebenfalls in SINA eingebunden werden um Implosion zu vermeiden.

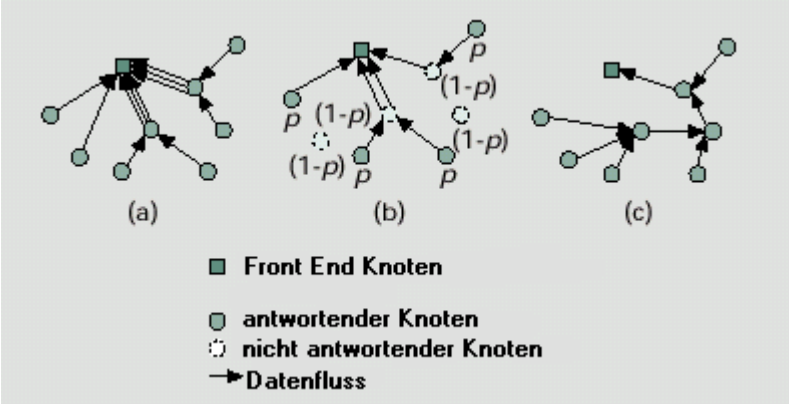


Abbildung 9 : (a) Das Implosion Problem. (b) Sampling Operation mit Wahrscheinlichkeit p. (c) Diffused Computation Operation. [SSJa00]

3.6.5 Ergebnisse in Tests

In einer Testreihe wurden die folgenden Verfahren untereinander verglichen :

Herkömmliches Flooding (Centralized), Sampling Operation (APR), Self Orchestrated Operation, Diffused Computation Operation, und die Mischformen Diffused + Sampling und Diffused + Self Orchestrated.

Dabei schneidet das Flooding natürlich am schlechtesten ab, wegen der vielen Buffer Overflows durch die begrenzte Bandbreite und durch die Kollisionen auf der MAC-Ebene. Das APR erzielt hier eine deutliche Verbesserung, ist aber schlechter als das Self-Orchestrated.

Die besten Ergebnisse erzielte das Diffused Computation, insbesondere kombiniert mit Sampling, da so sowohl Aggregationslogik in den Knoten ausgenutzt wird, als auch nur manche Knoten überhaupt antworteten, diese aber reichten um alle benötigten Informationen an den Front-End zu liefern.

Diese Ergebnisse werden anhand einer Tabelle verdeutlicht (Je höher der Anteil der empfangenen Paketen und je weniger MAC-Pakete dafür gebraucht wurden, desto besser) :

	Anteil empfangener zu erwarteten Paketen	Anzahl der MAC-Pakete pro Antwort
Flooding	29.8%	208.87
Self-Orchestrated	55.9%	138.17
APR	40.4%	164.19
Diffused Computation	99.2%	14.70
Diffused Computation + Sampling	99.8%	12.78
Diffused Computation + Self-Orch.	96.5%	14.24

Abbildung 10 : Tabelle mit dem Vergleich der Information Gathering Methods [SINA99]

3.7 Fazit

Zusammenfassend kann damit folgendes festgestellt werden :

- SINA bietet den Applikationen und dem Benutzer eine Möglichkeit zur Kontrolle von und Anfragen an Sensoren
- Benutzerfreundliche Schnittstelle mittels der Anfragesprache SQTL
- SINA vermeidet alle bereits erwähnten Probleme durch die Information Gathering Methods, insbesondere Diffused Computation Operation und SPIN
- SINA in Verbindung mit SPIN ist damit wohl die momentan beste Lösung zur Bewältigung der Aufgaben und Vermeidung der Probleme, die in einem Wireless Sensor Network anfallen

4 Power Awareness

Nachdem die betrachteten Verfahren SPIN und SINA hauptsächlich die Implosion- und Overlap-Problematik behandelten, werden in dem folgenden Abschnitt kurz Methoden zur Stromersparnis vorgestellt, das ein ernst zunehmendes Problem darstellt, da die Möglichkeit die Batterie jedes Sensors zu wechseln bei den meist sehr grossen Sensornetzwerken nicht geboten ist.

4.1 Dynamic Voltage Scaling

Hier wird die Tatsache ausgenutzt, dass der Energieverbrauch des Prozessors hauptsächlich von dessen Kernspannung abhängt. Folglich kann man Energie sparen, wenn man die Kernspannung und damit auch die Taktfrquenz des Prozessors herunter fährt. Das Dynamic Voltage Scaling [MBCS+01] bietet diese Möglichkeit und die Prozessoren in jedem Sensorknoten sind mittels des eingebauten Spannungswandlers in der Lage ihre Kernspannung eigenstädig zu ändern. Dies bietet sich allerdings nur an, wenn die Belastung des Prozessors niedrig ist oder die Verzögerung durch die verlangsamte Berechnung keine Rolle spielt. Man hat somit einen Tradeoff von Energie vs Verzögerung. In Tests wurde mit DVS eine Energieersparnis von 60% erzielt.

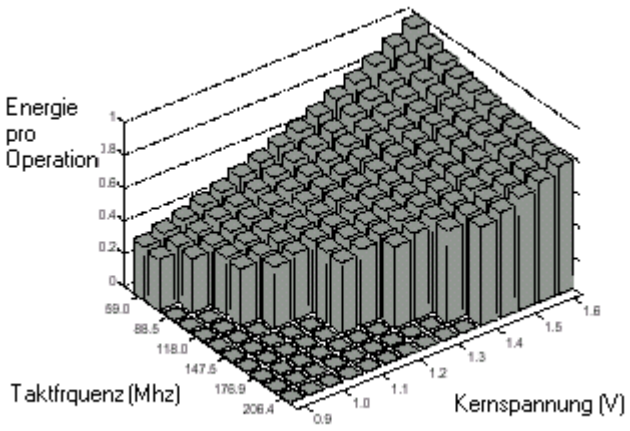


Abbildung 11 : Energieverbrauch eines SA-1100 Prozessors abhängig von der Kernspannung und der Taktfrequenz [MBCS+01]

4.2 Kommunikations-Hardware

Eine weitere sehr einfache Möglichkeit besteht im Abschalten der Kommunikations-Hardware während längerer Sende- bzw. Empfangspausen. Dadurch wird natürlich Energie gespart, allerdings auf Kosten von Verzögerung durch das „Hochfahren“ der Hardware.

4.3 Energie-effiziente Netzwerke

Nicht nur durch die Hardware selbst, sondern auch durch das Vorhandensein von Parallelität durch die vielen Sensoren und entsprechenden Betriebssystemen in den Knoten, kann Energie gespart werden.

4.3.1 Parallelität

Da ein Sensornetzwerk mit seinen vielen Knoten eine gute Basis für parallele Algorithmen bietet, kann auch durch parallele Verarbeitung von Daten Strom gespart werden, da je Knoten weniger Energie in gleicher Zeit verbraucht wird. In Abb. 12 ist dies anhand einer Fast Fourier Transformation verdeutlicht.

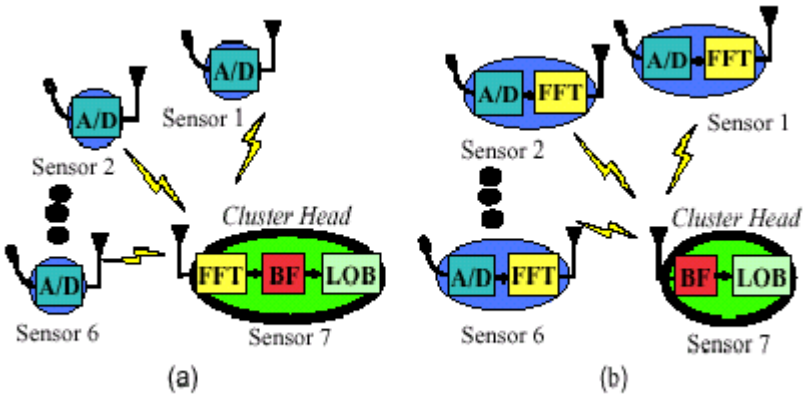


Abbildung 12 : (a) Sechs Sensoren senden ihre Daten an den Cluster Head. Dieser nimmt bei voller Leistungsfähigkeit die komplette FFT vor und transformeirt das Signal danach, bevor die fertigen Daten zum Endnutzer geschickt werden. (b) Die sechs Sensoren berechnen mit reduzierter Leistung eine partielle FFT und senden diese Daten an den Cluster Head, der die fertigen Ergebnisse zusammensetzt, transformiert und rausschickt. Die Energieersparnis liegt bei ca. 40% [MBCS+01]

4.3.2 Betriebssystem

Die oben erwähnten Shutdowns der Kommunikations-Hardware und das Herunterschalten der Kernspannung des Prozessors werden meist vom OS erledigt, da somit eine zentralere Steuerung der Energieressourcen möglich ist und das OS auch einen besseren Überblick über die jeweils geeigneten Energiesparmassnahmen hat.

4.4 Fazit

Durch das Steuern der Hardware, Algorithmen und Parallelität mittels Betriebssystem oder auch neuerdings mittels „Get Optimize Set“ Befehlen von APIs in der Programmierung von Applikationen, können sowohl Endbenutzer, wie auch Applikations-Programmierer eines Sensornetzwerks, genügend Einfluss auf die Stromsparfunktionen ausüben.

5 Eine Vision : Smart Kindergarten

5.1 Einführung, Aufgaben

Im Rahmen eines Projekts der UCLA [SMPo01] wurde ein Sensornetzwerk für den Einsatz in einem Kindergarten entwickelt, das nicht nur momentane Aktionen der Kinder überwachen sollte, sondern auch langfristige Prognosen über deren Lernverhalten liefern kann. Damit sind Ähnlichkeiten mit dem Classroom 2000 Projekt, bei dem Universitäts-Vorlesungen digital mit Hilfe von einem Liveboard, Video und Audio unterstützt werden, vorhanden.

Da die Entwicklung eines Kindes sehr stark von der Interaktion und Erforschung von Gegenständen abhängt, bietet es sich an Sensoren beispielsweise in die Spielzeuge einzubauen. Damit ist es dann möglich Aussagen der Art „Kind A hat Spielzeug B am Ort C entdeckt und mit Spielzeug D in Raum E kombiniert“. Diese Aussagen werden vom Netzwerk an ein „Background-System“ übermittelt und dort gesammelt und ausgewertet. Somit erhofft man sich Prognosen über Lernverhalten der Kinder und Optimierung der Ausstattung des Kindergartens.

5.2 System Architektur

Die System Architektur des Smart Kindergarten besteht hauptsächlich aus den Komponenten die auch schon in dieser Arbeit erörtert wurden. Viele kleine Sensoren in Spielzeugen um die Kinder und das Spielzeug zu identifizieren, in den Räumen für die Ortserkennung (GPS und lokale Bestimmung), aber auch Mini Kameras und Microphone, die bei bestimmten Aktionen aktiviert werden. Diese kommunizieren auf Bluetooth Basis [BLUE] und übermitteln ihre Daten an Cluster Heads, zu denen auch die Kameras wegen ihrer hohen Bandbreite gehören, die den Backbone eines 802.11b WLANs bilden. Dies steht wiederum mit einem traditionellen LAN in Verbindung, in dem Datenbank- und Applikationserver stehen um die Daten zu sammeln und auszuwerten (Abb. 13).

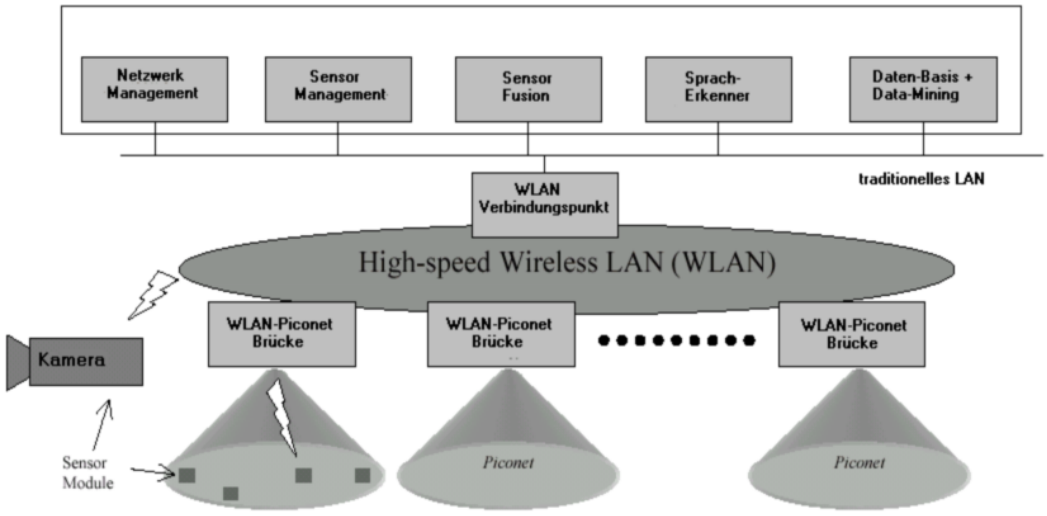


Abbildung 13 : Das Smart Kindergarten Netzwerk, bestehend aus Piconets (Bluetooth-Sensoren) + Kameras, dem WLAN und dem „Background Network“ [SMPo01]

5.3 Fazit + Umsetzung der erörterten Techniken

Nicht nur die eigentliche Architektur des Smart Kindergarten, sondern auch die verwendeten Middleware und Netzwerk-Protokolle stimmen größtenteils mit den in dieser Ausarbeitung vorgestellten Techniken überein bzw sind so zu erwarten, da das Projekt noch in der Planung steht :

- Das Bilden einer Hierarchie von Clustern incl. Cluster Heads an der Spitze (SINA)
- Verwenden von Meta Daten, eines Namensdienstes, einer Anfragesprache und Informations-Beschaffungsmaßnahmen um Bandbreite zu sparen und dem Implosion- und Overlap-Problem entgegenzutreten (SPIN + SINA)
- DVS und allgemein stromsparende Maßnahmen der Hardware
- Ausnutzen paralleler Architektur bedingt durch die grosse Anzahl an Sensoren
- APIs um das Verhalten des Netzes und auch dessen Stromersparnis zu steuern

Zusätzlich bietet das Smart Kindergarten Netzwerk die Anbindung an ein normales LAN, um dort mit Datenbank-, Applikations- und Datamining-Methoden, die speziell für die Sensor-Umgebung geeignet sind, Auswertungen fahren zu können.

Somit ist das Smart Kindergarten Netzwerk ein gutes Beispiel, wie sowohl Wireless Sensor Networks selbst, als auch Technken, die die Probleme, die in diesen Netzwerken auftreten können, lösen, in der Praxis umgesetzt werden könnten.

6 Zusammenfassung

In der vorliegenden Ausarbeitung wurden Wireless Sensor Networks incl. deren Aufbau, Nutzen, vorhandenen Probleme und zu den Problemen passende Lösungsansätze vorgestellt. Insbesondere wurden die Middleware Services SPIN bzw. SINA und Stromsparmaßnahmen der Hardware erörtert. Anhand des Beispiels des Smart Kindergartens wurde gezeigt wie diese Techniken auch in der Praxis Einsatz finden bzw. Einsatz finden könnten.

7 Literatur

[MBCS+01] Rex Min, Manish Bhardwaj, Seong-Hwan Cho, Eugene Shih, Amit Sinha, Alice Wang, Anantha Chandrakasan : „Low-Power Wireless Sensor Networks“ , MIT, Department of EECS , Paper, Januar 2001, www.mit.edu

[HKBa99] Wendy Rabiner Heinzelman, Joanna Kulik, Hari Balakrishnan : „Adaptive Protocols for Information Dissemination in Wireless Sensor Networks“, MIT, Fifth ACM/IEEE Mobicom Conference, Seattle, WA, August 1999, www.mit.edu

[SSJa00] Chien –Chung Shen, Chavalit Srisathapornphat, Chaiporn Jaikaeo : „Sensor Information Networking Architecture and Applications“, University of Delaware, International Workshop on Pervasive Computing, Toronto, Canada, August 2000, www.udel.edu

[SMPo01] Mani Srivastava, Richard Muntz, Miodrag Potkonjak : „Smart Kindergarten Sensor-based Wireless Networks for Smart Developmental Problem-solving Environments“, ACM Sigmobile 7/01, Rome, Italy, Pages 132-138, www.ucla.edu

[BLUE] www.bluetooth.com