# Wireless Sensor Networks

Alexander Holzinger

**Teleseminar Ubiquitous Computing WS 2001/2002**

Universität Mannheim

Praktische Informatik IV

Prof. Dr. W. Effelsberg

# Gliederung

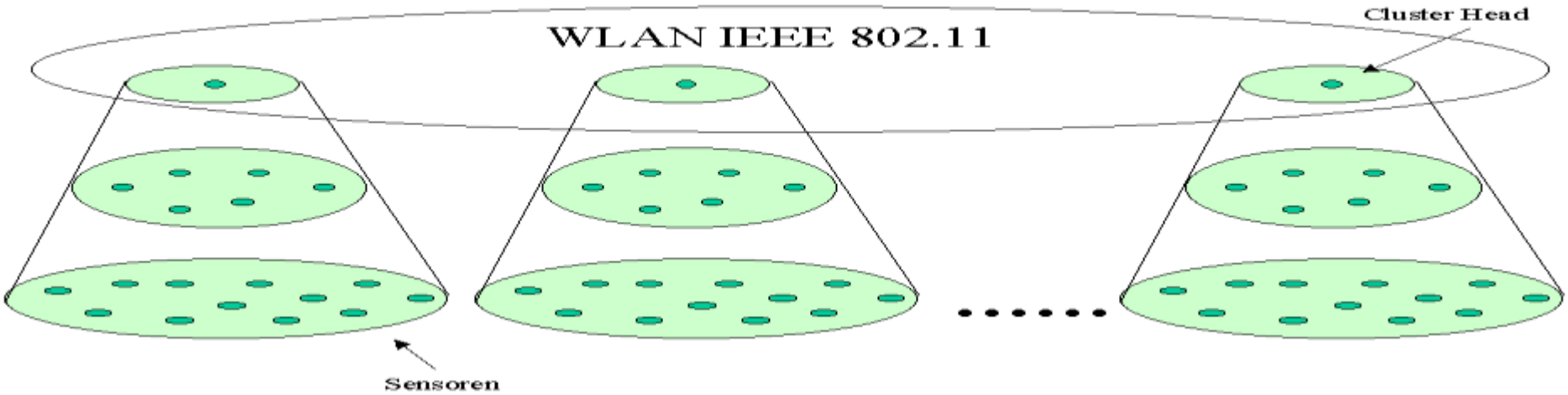
# 1 Einführung

- Wie ist ein Sensor Netzwerk aufgebaut ?
- Welche Aufgaben erfüllt es ?
- Welche Probleme existieren in den Sensor-Netzwerken ?
- Welche Lösungsansätze gibt es ?

# 1.1 Aufbau und Aufgaben eines Wireless Sensor Network

Ein **Wireless Sensor Network** ist :

- Zusammenschluss von vielen kleinen, batterie-betriebenen Sensorknoten
- sammelt und verarbeitet Messdaten und Informationen, die dem Benutzer zur Verfügung gestellt werden
- Informationen sind von kurzfristiger Natur (Messdaten, Video und Audio), oder können langfristig genutzt werden (Datenmanagement, Data-Mining)
- Knoten topologisch regional und hierarchisch geordnet, mit “Cluster Heads” an der Spitze
- Kommunikation auf unterer Ebene mit z.B. Bluetooth, auf oberer mit IEEE 802.11



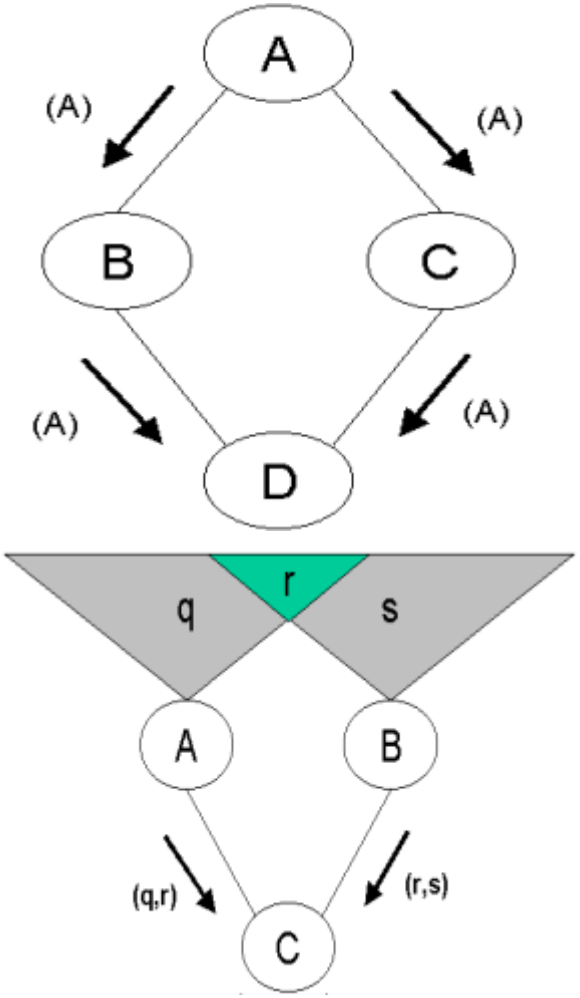
# 1.2 Probleme in Sensor-Netzwerken

Probleme bedingt durch Topologie und Einsatzort der Netzwerke :

- **Response-Implosion-Problem :**
  doppelt ankommende Nachrichten in den Knoten als Folge von Flooding ( ⟹ Bandbreite, Rechenleistung, Energie)
  Lösungsansatz durch SPIN und SINA (Abschnitt 2 und 3)

- **Overlap-Problem :**
  Überlappen der Einzugsgebiete der Sensoren
  ( ⟹ siehe Implosion)
  Lösungsansatz durch SPIN und SINA (Abschnitt 2 und 3)

- **Limited-Power-Problem :**
  beschränkte Energiereserven, bedingt durch den Betrieb mit Batterien
  Lösungsansatz durch DVS, Energy-efficient Networks (Abschnitt 4)



# 2. Das SPIN-Protokoll

- **SPIN = Sensor Protocol for Information via Negotiation**

- 1999 am MIT als **Middleware**-Protokoll entwickelt (d.h. für Applikationen über Bibliotheken und APIs zu erreichen, steht über der Transport-Schicht)

- verhindert Implosion- und Overlap-Probleme durch Verwenden von **Meta-Daten** und **Kommunikation vor** dem Versenden der Anwendungsdaten (2.1)

# 2.1 Meta-Daten, Nachrichten und Protokoll-Ablauf

**Meta-Daten :**

- Kommunikation von Nachbarknoten untereinander vor dem Versenden der Nutzdaten
  ⇒ sich darüber klar sein, ob ein Knoten die Daten, die ein anderer verschickt, überhaupt braucht bzw. schon hat

- Meta-Daten = Konstruktion, um die Nutzdaten einheitlich beschreiben und benennen zu können

- **SPIN** : Format der Meta-Daten anwendungsabhängig, z.B. Meta-Data (x,y) = Messdaten y des Sensors x bzw. (x,y,z) = Kamerabilder der Region (x,y) mit Orientierung z

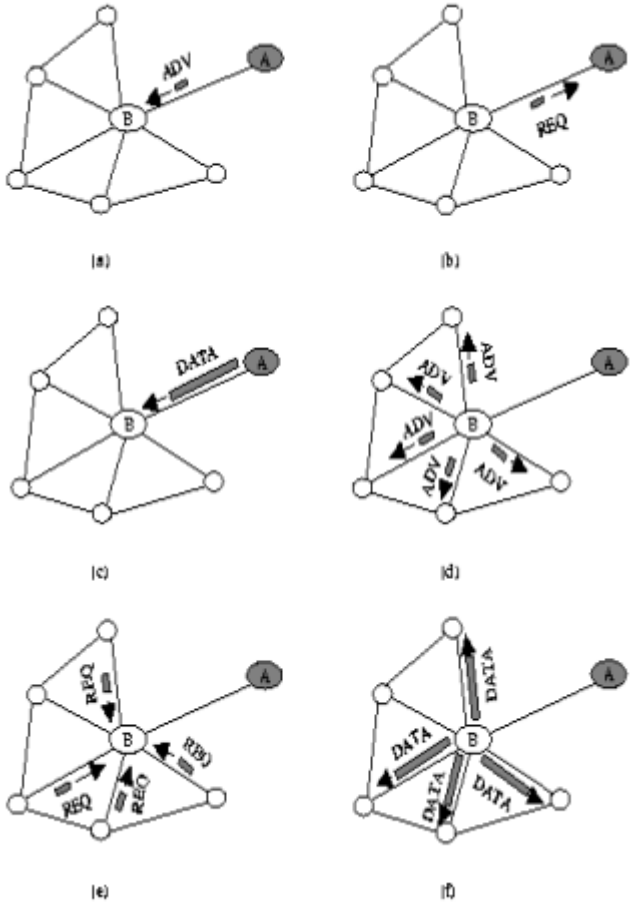
- Aufwand für die zusätzliche Kommunikation durch die Meta-Daten ist sehr viel geringer als unnötiger Versand von doppelt vorhandenen Nutzdaten

⇒ Vermeidung von Implosion und Overlap bzw. effizientere Nutzung von Energie und Bandbreite

# 2.1 Meta-Daten, Nachrichten und Protokoll-Ablauf

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

- **ADV** : new data advertisement : dem Knoten liegen neue Nutzdaten vor, die er Versenden möchte
- **REQ** : request for data : verschickt von einem Knoten, der neue Daten haben möchte
- **DATA** : data message : enthält aktuelle Nutzdaten




SPIN ist ein einfaches **3-Stufen Handshake Protokoll**, bei dem jede Stufe einer Nachricht entspricht :

- **Stufe 1 :** Das Vorliegen neuer Daten wird verbreitet (ADV)
- **Stufe 2 :** Nachbarknoten fordern diese Daten an (REQ)
- **Stufe 3 :** Daten werden versendet (DATA)

## 2.2 Testergebnisse

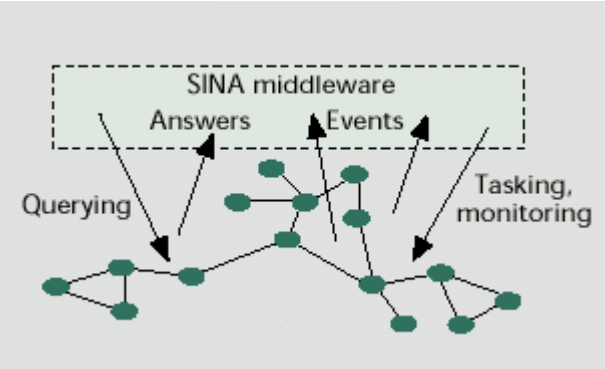
Vergleich von **SPIN** mit **Flooding** bzw. einem **„perfekten Protokoll“** (Test am MIT) :

- SPIN braucht wegen der zusätzlichen Kommunikation länger als Flooding um Daten zu verteilen (Faktor 1.5)
- SPIN verbraucht aber deutlich weniger Bandbreite (Faktor 1.75) und Energie (Faktor 4) als Flooding
- SPIN kommt in der Performance nahe an das Ideal-Protokoll

⟹ **SPIN** gut geeignet für Wireless Sensor Networks

# 3 SINA

- **SINA = Sensor Information Networking Architecture**
- wurde 2000 von der University of Delaware entwickelt
- bietet nicht nur ein Protokoll, sondern eine komplette Architektur in der Middleware
- Dienste wie das Verarbeiten von Anfragen, Kommandos, Antworten und Veränderungen des Sensor Netzwerks



# 3.1 SINA-Architektur

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

- **Hierarchical Clustering der Sensorknoten :**

  Aufteilung der Knoten in Blattknoten, Knoten der mittleren Ebene und Cluster Heads

  Vorteil : nicht alle Knoten müssen alle Aufgaben übernehmen
- **Attribute-Based Naming :**

  attribut-basierter Namensdienst, ähnlich der Meta Daten in SPIN

  Vorteil : Aggregation und somit effizientere Ausnutzung der Bandbreite
- **Location Awareness :**

  Ermittlung des Aufenthaltsorts der Knoten z.B. über GPS
- **Sensor Query and Tasking Language (SQTL) :**

  Schnittstelle zwischen Applikation und SINA-Middleware, SQL ähnlich, für Zugriffe auf Hardware, Ort und Kommunikation der Knoten

# 3.2 Information Gathering Methods

Anfragen und Befehle, die mittels SQTL an die Middleware gestellt werden, gehen an das SSE (Sensor Execution Environment), das auf jedem Knoten läuft.

Das SSE verwendet dabei verschiedene Techniken zur Informationsverarbeitung :

- **Sampling Operation** (auch **APR** genannt) :

  Knoten entscheidet anhand einer Wahrscheinlichkeit p, ob er Daten sendet oder nicht.

- **Self Orchestrated Operation :**

  Hier sendet jeder Knoten, allerdings mit einer Verzögerung d.

- **Diffused Computation Operation :**

  Aggregations-Logik in jedem Knoten vorhanden, die dem Knoten sagt :

  1) ob Daten in Richtung Front-End zu versenden sind oder nicht

  2) wie die Daten zu aggregieren sind

⟹ erhöhte Verabeitungszeit in den Knoten bedingt durch die Logik, aber :

**Bandbreite wird eingespart !**

## 3.3 Testergebnisse

In Tests wurden verschiedene Information Gathering Methods untereinander verglichen:

- ohne besondere Technik (Flooding) wurden die schlechtesten Ergebnisse erzielt
- APR ist besser als Flooding, aber schlechter als Self-Orchestrated
- die besten Ergebnisse durch **Diffused Computation,** insbesondere kombiniert mit **APR**, da :

  1) **Aggregationslogik** in den Knoten ausgenutzt wird

  2) nur manche Knoten mit **Wahrscheinlichkeit p** überhaupt antworten

  3**)** dies reicht aber, um alle benötigten Informationen an den Front-End zu liefern

# 3.3 Testergebnisse

| | 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 |

Fazit : SINA bietet somit eine benutzerfreundliche Umgebung zur Kontrolle der Sensoren und vermeidet Implosion- und Overlap-Probleme durch Verwendung von Aggregation (Meta-Daten) und Informationsbeschaffungs-Methoden

# 4 Power Awareness

Neben der Implosion- und Overlap-Problematik, existieren auch **Energie-Probleme** in den einzelnen Knoten :

- nur begrenzte Energiereserven aufgrund von Batterie-Betrieb
- Auswechseln der Batterien bei so vielen Knoten keine Option darstellt

⇒ Energiesparmaßnahmen in der **Hardware und Software** der Knoten sind unabdingbar

**Hardware :**

- **Dynamic Voltage Scaling**

  Die Kernspannung und damit die Taktfrequenz des Prozessors wird heruntergefahren

  ⇒ Tradeoff zwischen Energieersparnis und Leistung

# 4 Power Awareness

- **Kommunikations Hardware**

  Abschalten der Funk-Transceivers während längerer Sende- bzw. Empfangspausen

  ⟹ Tradeoff zwischen Energieersparnis und Verzögerung durch „Hochfahren“ der Hardware

**Software :**

- **Parallelität**

  Intelligente Ausnutzung der parallelen Architektur des Netzes in den Software-Algorithmen der Knoten ⟹ jeder Knoten verbraucht weniger Energie in derselben Zeitspanne um dasselbe Ergebnis zu erzielen, da jeder Knoten weniger rechnen muss

- **Betriebssystem**

  Das OS hat den besten Überblick über alle Ressourcen und kann somit effiziente Energie-sparmaßnahmen durchführen

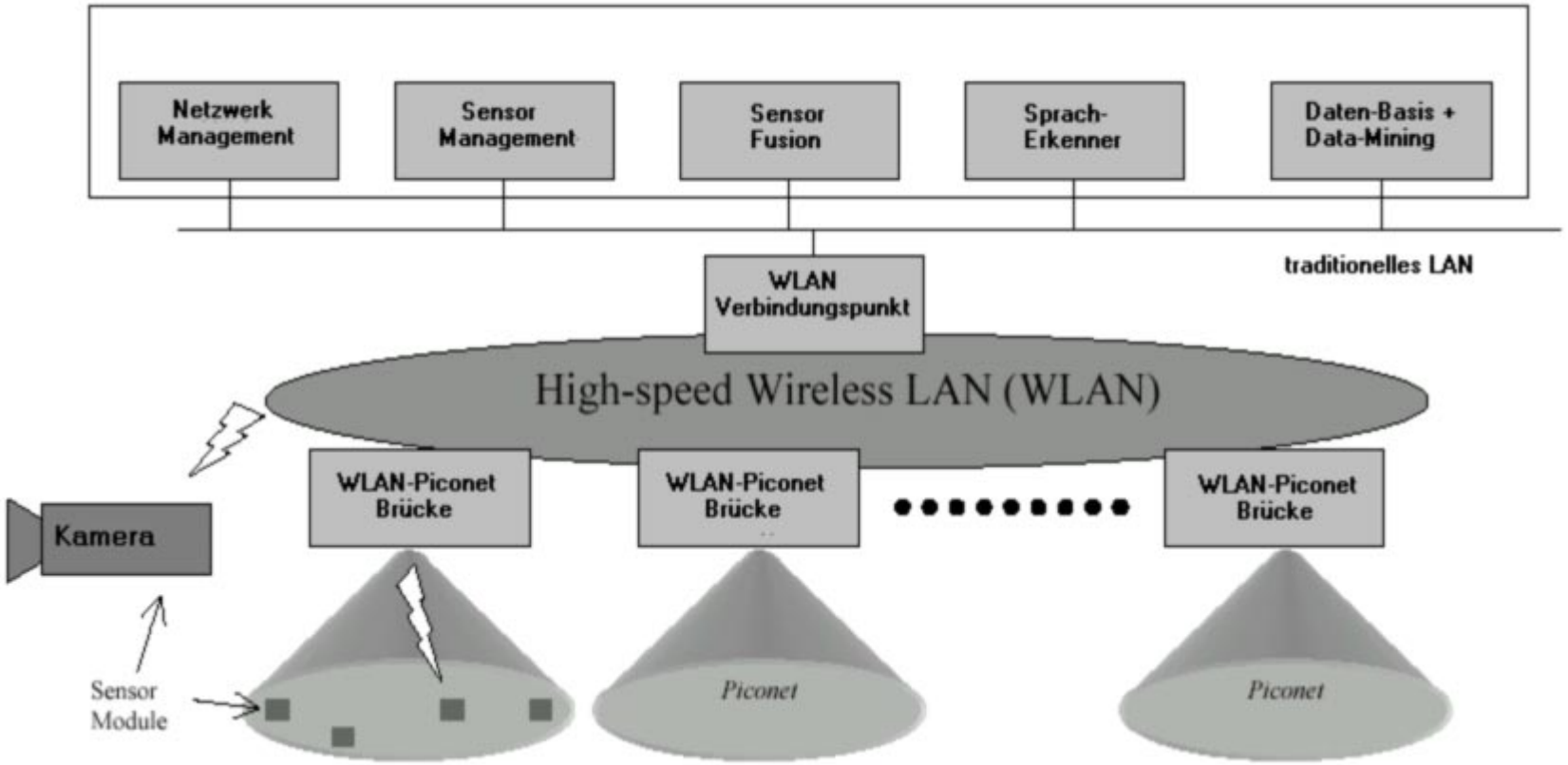
# 5 Projekt : Smart Kindergarten

- aktuelles Projekt der UCLA
- Sensornetzwerk für den Einsatz in einem Kindergarten
- nicht nur **kurzfristige** Aktionen der Kinder werden überwacht, sondern auch **langfristige** Prognosen über deren Lernverhalten sollen möglich sein
- Sensoren werden in Spielzeuge und Räume installiert um z.B. Aussagen der Art "Kind A hat Spielzeug B am Ort C entdeckt und mit Spielzeug D in Raum E kombiniert" zu treffen
- Aussagen werden vom Netzwerk an ein "Background-System" übermittelt und dort gesammelt und ausgewertet

  ⇒ Prognosen über Lernverhalten der Kinder und Optimierung der Ausstattung des Kindergartens u.U. möglich

# 5.1 System Architektur

Aufbau des Smart Kindergarten Netzwerks :



# 5.1 System Architektur

**3 Hauptbestandteile :**

- **kleine Sensoren** in Spielzeugen und Räumen, Kommunikation untereinander über **Bluetooth**

- die **Cluster Heads** bilden den Backbone mit einem **802.11b WLAN** und stehen in Verbindung zu einem **traditionellen „Kabel“ - LAN**

- in diesem **„Background System“** werden die Informationen der Sensoren gespeichert und verarbeitet (Data-Mining, Spracherkennung und Management der Sensoren sollen über APIs der Middleware möglich sein )

# 5.2 Fazit

Architektur und verwendete Techniken des Smart Kindergarten Netzwerks sollen größtenteils mit den hier vorgestellten Techniken übereinstimmen und somit auch die Probleme, die für Wireless LANs gelten, beseitigen :

- Das Bilden einer Hierarchie von **Clustern incl. Cluster Heads** an der Spitze (SINA)
- **Meta Daten**, Namensdienstes, Anfragesprache und **Informations-Beschaffungsmaßnahmen** um Bandbreite zu sparen und dem Implosion- und Overlap-Problem entgegenzutreten (SPIN + SINA)
- **DVS** und allgemein stromsparende Maßnahmen der **Hardware und Software** werden in den Knoten zum Einsatz kommen (Power Awareness)
- **APIs** um das Verhalten des Netzes und auch dessen Stromersparnis zu steuern (SINA)

⟹ **Smart Kindergarten** als Referenz für Sensornetzwerke und was mit ihnen alles möglich ist bzw. sein wird