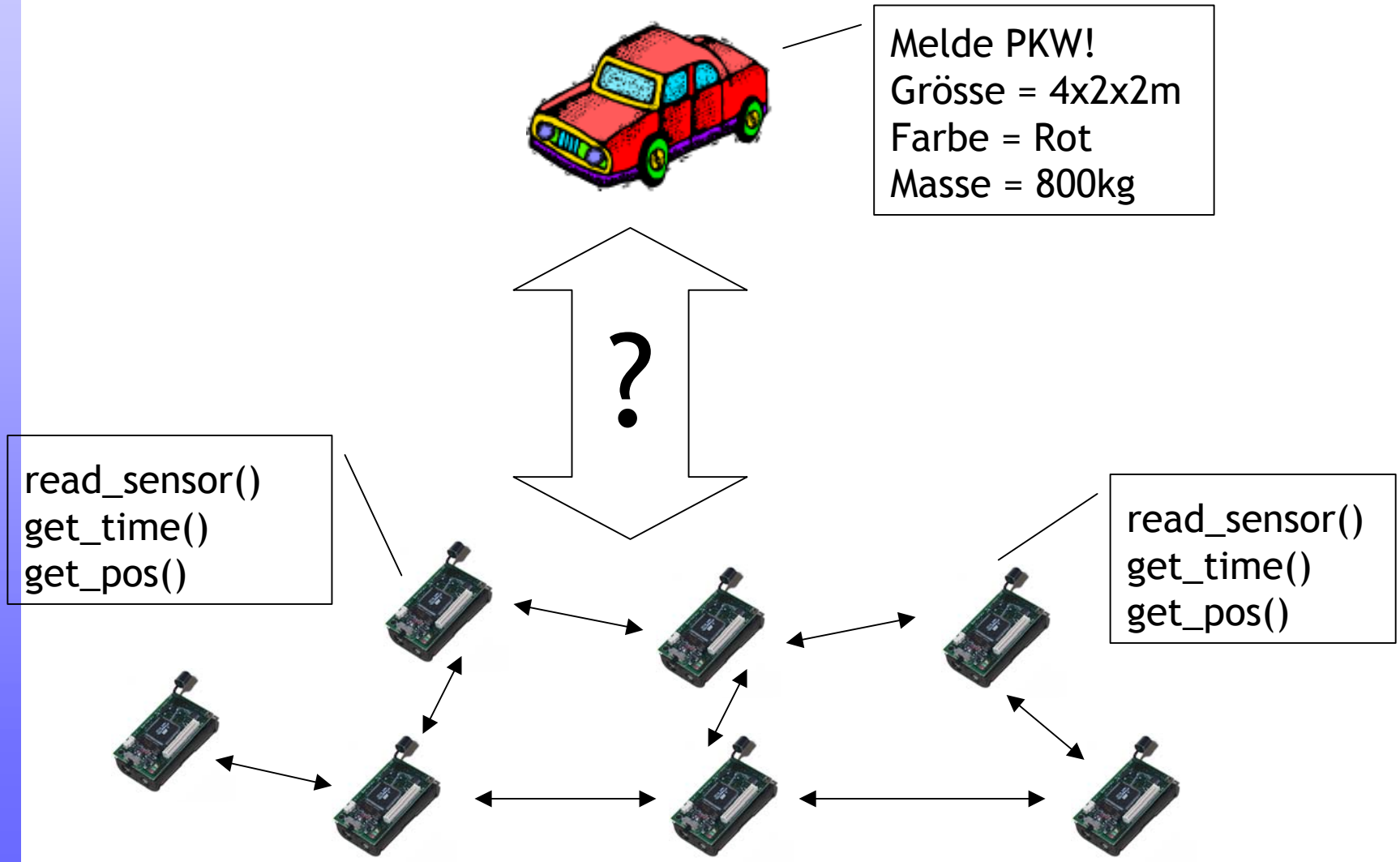
Programmierung von Sensornetzen

Kay Römer
ETH Zürich
Switzerland

The Gap


Melde PKW!
Grösse = 4x2x2m
Farbe = Rot
Masse = 800kg
?
read_sensor()
get_time()
get_pos()
read_sensor()
get_time()
get_pos()

Beobachtungen

- Problembeschreibung
  - Hohes Abstraktionsniveau
  - „problemorientiert“
- Sensornetz
  - Niedriges Abstraktionsniveau
  - „systemorientiert“, verteiltes System
- System Software, Middleware
  - Kann Abstraktionsniveau des Sensornetzes heben
  - Unterstützung geeigneter Programmierparadigmen
- Ziel
  - Automatismen zur „Annäherung“ von Problem und Sensornetz

Programmier-Paradigmen

- Klassische Ansätze in verteilten Systemen
  - Beispiele: Verteiltes Objektmodell, Events, Agenten, Shared Information Spaces
- Sensornetze
  - Datenzentrisch
  - Verteilte Signalverarbeitung
  - Inhärente Kollaboration vieler Knoten
  - „Emergent behavior“

State of the Art

- Individuelle Programmierung einzelner Sensorknoten
  - Manuelle Anpassung der Abstraktionsniveaus
- Erste Ansätze für höhere Programmier-abstraktionen und entsprechende Middleware
  - TinyDB (Datenbanken), UCB
  - SensorWare (Mobile Agenten), UCLA
  - DSWare (Events), Virginia
  - EnviroTrack, Virginia
  - Abstract Regions, Harvard
  - SWARMS, TU Braunschweig
  - Role Assignment, ETHZ

TinyDB [OSDI 02]

- Sensornetz wird als verteilte Datenbank interpretiert
- SQL-artige Queries
  - SELECT, GROUP BY, Aggregation
  - Streaming Queries (periodische Ausführung)
- Query Auswertung
  - Initiiert durch Senke
  - Spannbaumkonstruktion durch Flooding
  - Datenquellen senden entlang Senke
  - Innere Knoten aggregieren

TinyDB: Beispiel

- Räume mit Durchschnittstemperatur > 10° alle 30 Sekunden

```
SELECT AVG(temp), room FROM sensors
    GROUP BY room
    HAVING AVG(temp) > 10
    EPOCH DURATION 30s
```

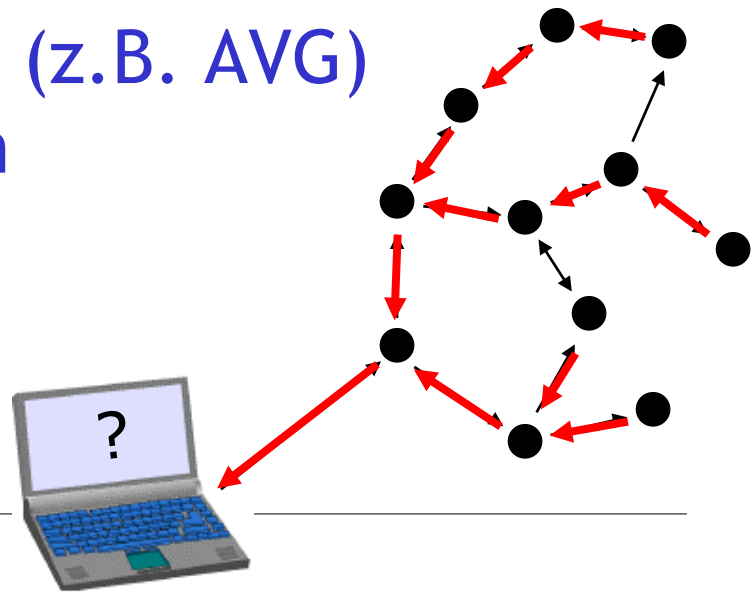
Gruppierung der Sensoren nach Raum

Auswahl von Gruppen mit ∅ > 10

Alle 30 Sekunden

TinyDB

- Ausführung einer Anfrage
  - Sensornetz wird mit der Anfrage geflutet
    - Spannbaumkonstruktion
  - Periodisches (EPOCH) Auslesen der Sensoren
  - Innere Knoten warten auf Daten von Kindern und aggregieren (z.B. AVG) mit eigenen Sensordaten
  - Daten an Vater im Spannbaum schicken



TinyDB: Antwort

- Antwort-Nachrichten
  - Für jede Gruppe (GROUP BY) ein Eintrag pro Nachricht
  - Eintrag für Gruppe abhängig vom Aggr. Operator
    - MIN: Minimum unter den Kindern
    - AVG: Summe der Werte, Anzahl Werte
    - MEDIAN: Werte aller Kinder und Kindeskinder (!)
- End-Auswertung
  - Auswahl der Gruppen (HAVING)
  - Auswertung der aggr. Werte abhänging vom Operator
    - MIN: Minimum
    - AVG: Summe / Anzahl
    - MEDIAN: Wert in der Mitte der sortierten Werte

TinyDB: Eigenschaften

- Hohes Abstraktionsniveau
  - Macht Verteiltheit transparent
- Mächtigkeit
  - Snapshots zu bestimmten Zeitpunkten
  - Keine explizite Unterstützung von zeitlichen / örtlichen Bezügen
  - Vordefinierte Operatoren
- Effizienz
  - Globales Flooding
  - Problemunabhängige Netztopologie

SensorWare [MobiSys 03]

- Inspiriert von mobilen Agenten
- Zustandbehaftete Skripte replizieren sich im Netz
  - Spezielle Kommandos: spawn, replicate
  - Zugriff auf lokale Sensoren
  - Kommunikation mit remote Skripten
- Laufzeitumgebung auf Knoten
  - Erweiterter Tcl Interpreter
  - Skript Ausführung
  - Ressourcenüberwachung

SensorWare: Beispiel

- Maximale Temperatur

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

SensorWare: Eigenschaften

- Abstraktionsniveau
  - Programmierung individueller Knoten
  - Replikation von Skripten
- Mächtigkeit
  - Beliebige verteilte Algorithmen
- Effizienz
  - Keine globalen Netzstrukturen
  - Effizienz interpretierter Skripte?

DSWare [IPSN 03]

- Verteiltes Event System
- Sensorknoten generieren Basis-Events
  - Bei Auftreten bestimmter Sensorwerte
- Erkennung bestimmter Event-Muster
  - „Composite Events“
  - Menge von Basis-Events in Zeitfenster
  - Einzelne Events gewichtet

DSWare: Beispiel

- Explosion, wenn Knall und Hitze und Blitz innerhalb kurzer Zeit
  - Basis-Events
  - Hitze ist wichtiger als Knall und Blitz

  6 B(Hitze) + 3 B(Blitz) + 3 B(Knall) >= 9

DSWare: Eigenschaften

- Abstraktionsniveau
  - Ortstransparenz bei Event-Auslieferung
  - Komplexere Event-Muster?
  - Glue Code notwendig
- Mächtigkeit
  - Composite Events nur ein Hilfsmittel
  - Im wesentlichen durch Glue Code bestimmt
- Effizienz
  - Kaum zusätzlicher Overhead

EnviroTrack [ICDCS 04]

- Mobiles Objekt wird durch veränderliche Gruppe von Sensoren in der Nähe des Objektes beobachtet
- „Beobachtergruppe" wird als adressierbares Element zur Verfügung gestellt
  - Anheften von Zustand, Aktionen
- Laufzeitsystem unterstützt
  - Dynamische Gruppenverwaltung
  - Migration von Zustand

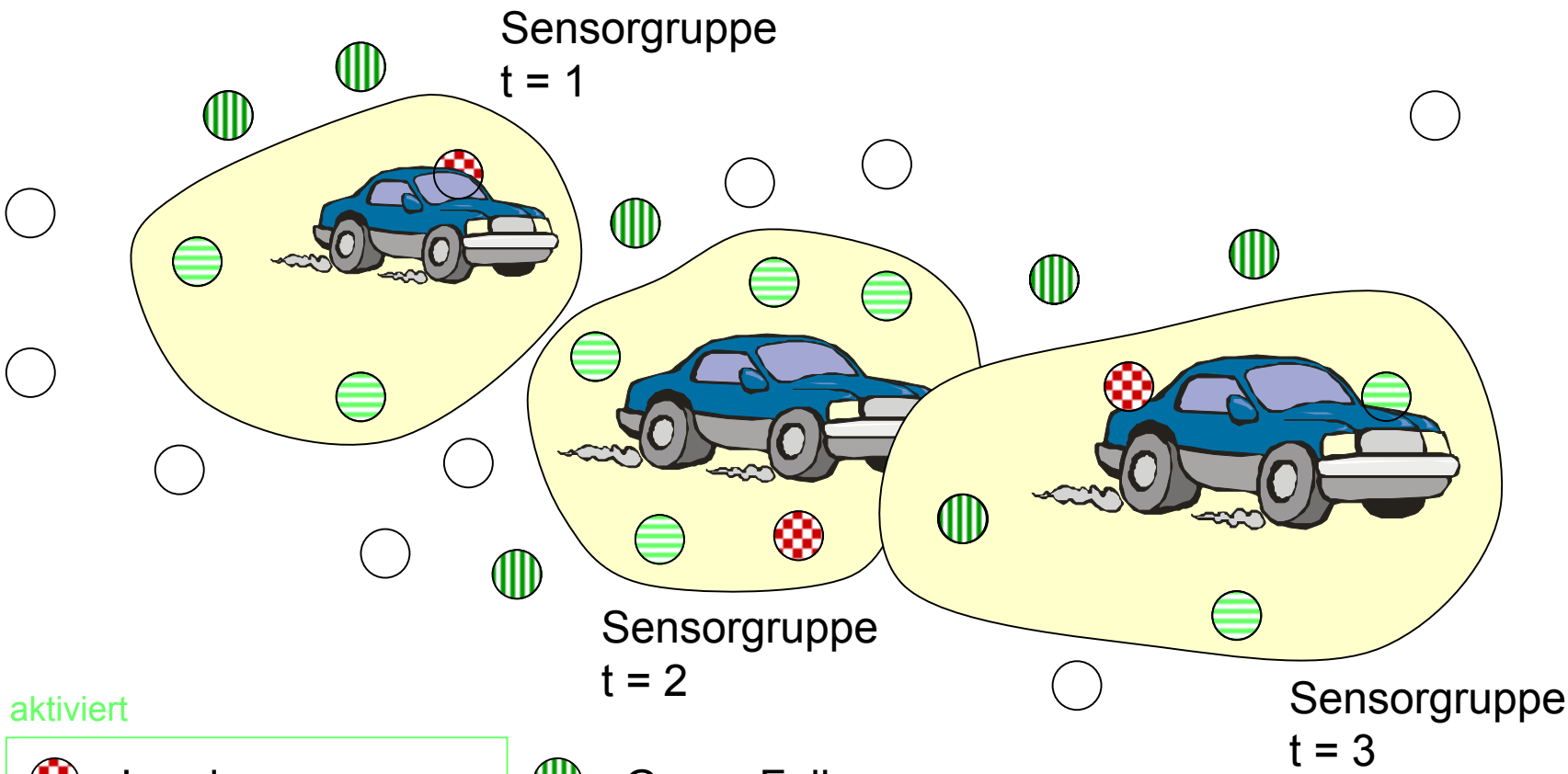
EnviroTrack: Beispiel

- Verfolgen eines Fahrzeuges

```
begin context tracker
    activation : pkw_erkannt()
    loc : avg (position) confidence = 2, freshness = 1s

    begin object reporter
        invocation : TIMER(5s)
        report_function() {
            /* loc an Basisstation senden */
        }
    end object
end context
```

EnviroTrack: Beispiel



EnviroTrack: Eigenschaften

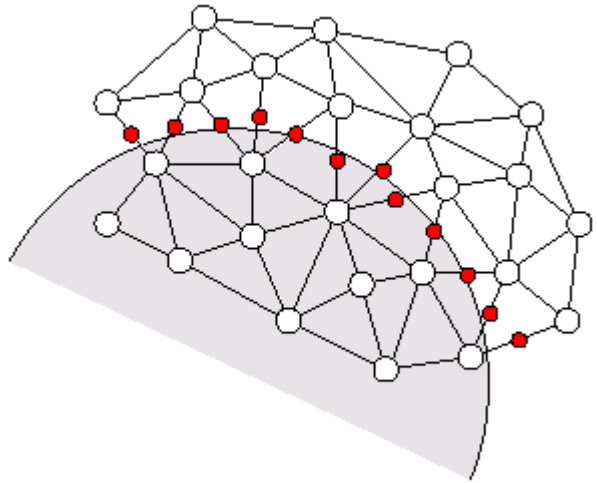
- Abstraktionsniveau
  - Abstrahiert von Verteiltheit und Objektmobilität
- Mächtigkeit
  - Speziell für Beobachtung mobiler Objekte
- Effizienz
  - Precompliation
  - Gruppenverwaltung?

Abstract Regions [NSDI 04]

- Region: Gruppe von Knoten mit geographischem od. topologischen Zusammenhang
  - Alle Knoten im Abstand d
  - Alle Knoten im Spannbaum
  - Planarer Graph
- Knoten in einer Region haben Zugriff auf gemeinsam benutzte Variablen
  - Vgl. Tupelspace
  - putvar(key,val), get(key,node)
- Datenaggregation auf den Variablen
  - reduce(op,v,res)

AR: Beispiele

- Contour Finding



```
location = get_location();
region = apmesh_region.create();
region.putvar(loc_key, location);
while (true) {
    reading = get_sensor_reading();
    region.putvar(reading_key, reading);
    if (reading > threshold) {
        foreach (nbr in region.get_neighbors()) {
            rem_reading = region.getvar(reading_key, nbr);
            if (rem_reading <= threshold) {
                rem_loc = region.getvar(loc_key, nbr);
                contour_point = midpoint(location, rem_loc);
                send_to_basestation(contour_point);
            }
        }
    }
    sleep(periodic_delay);
}
```

AR: Eigenschaften

- Abstraktionsniveau
  - Gruppenverwaltung, gemeinsame Variablen, Datenaggregation
  - Umfangreicher Glue Code notwendig
- Mächtigkeit
  - Grundlegende Abstraktion in vielen WSN Applikationen
- Effizienz
  - Ausnutzung von Lokalität

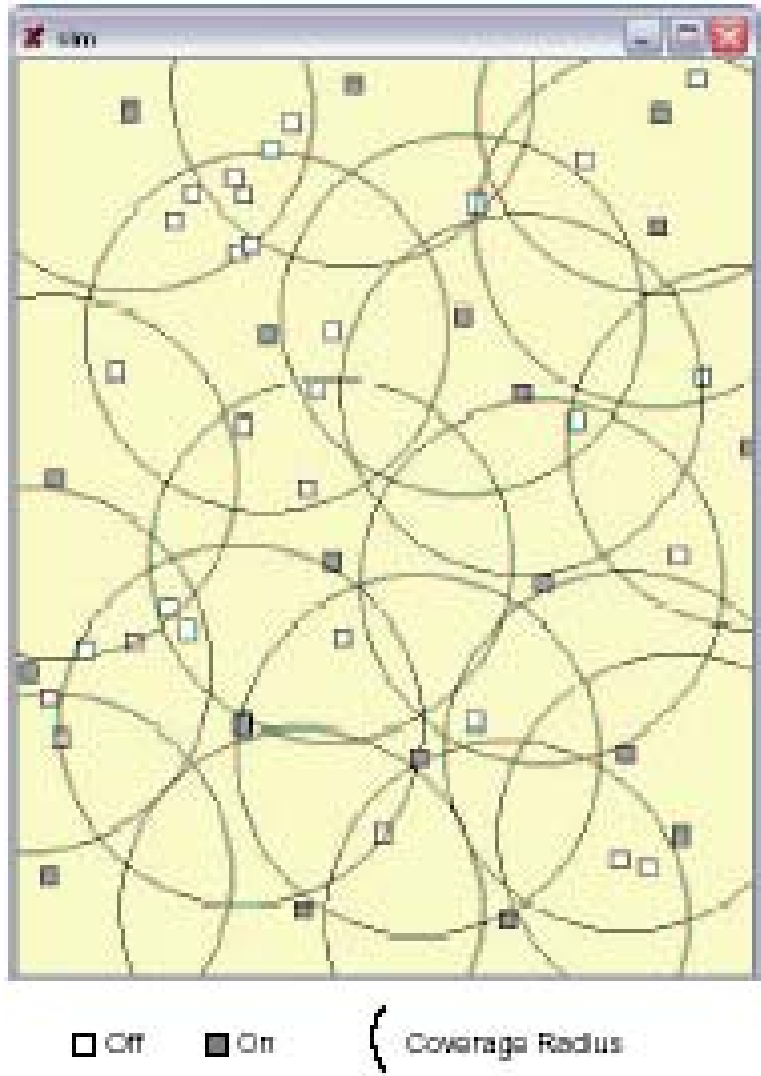
Role Assignment [SIGOPS EW 04]

- Sensorknoten unterscheiden sich in Ihren Eigenschaften
  - Sensoren, Ressourcen, Netzumgebung, Position, ...
- Knoten sollen Rollen übernehmen abhängig von ihren Eigenschaften
- Framework zur automatischen Zuweisung von Rollen
  - Benutzerdefinierte Rollen
  - Regeln für die Zuweisung
  - Lokaler Dienst für Knotenparametern
  - Vert. Algorithmus für dynamische Rollenzuweisung

RA: Coverage

- Wähle wenige Knoten, so dass Gebiet abgedeckt wird

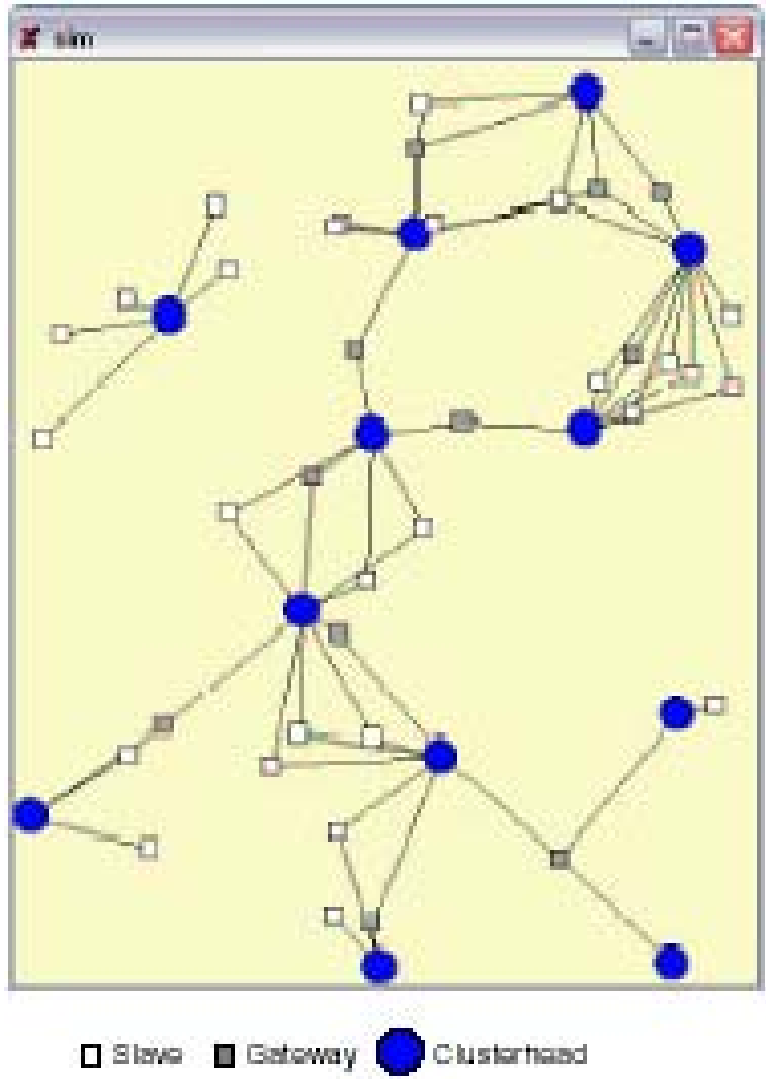
```
ON :: {
    sensor == temp &&
    battery >= threshold &&
    count (2 meters) {
            role == ON
    } <= 1
}
OFF :: else
```



RA: Clustering

- Bilde 1-Hop Cluster

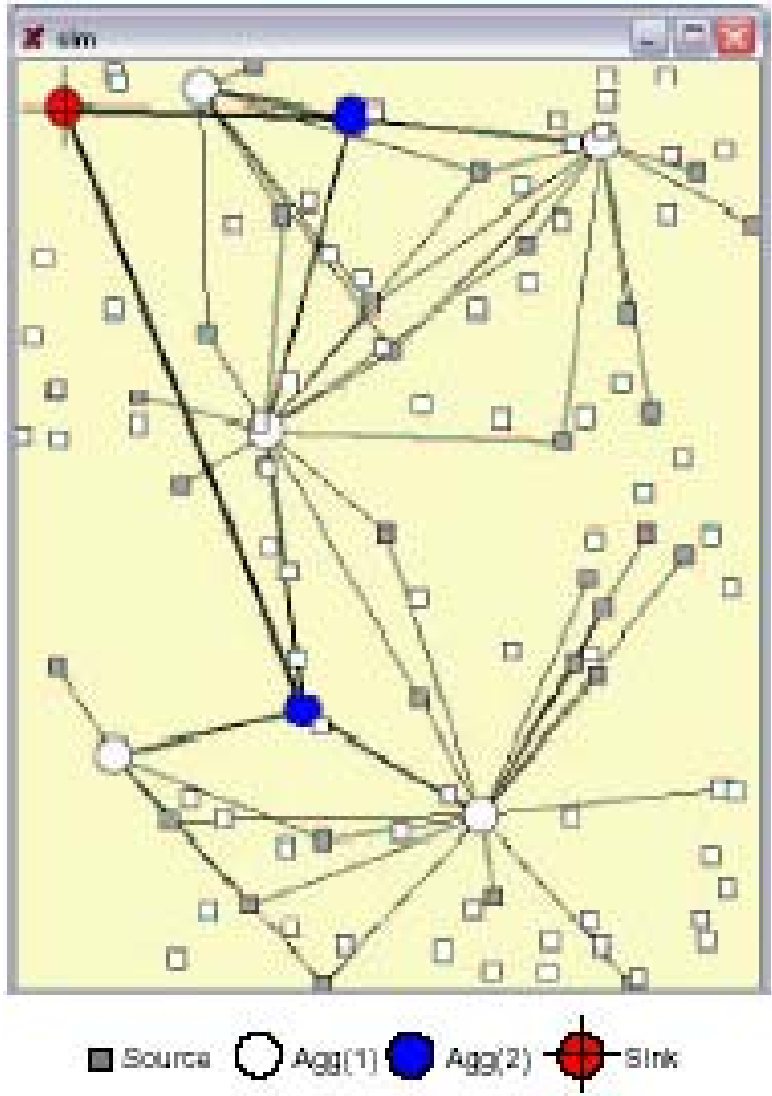
```
CLUSTERHEAD :: {
    count(1 hop) {
        role == CLUSTERHEAD
    } == 0
}
GATEWAY(c1,c2) :: {
    retrieve(1 hop, 2) {
        role == CLUSTERHEAD
    } == (c1,c2) &&
    count(2 hops) {
        role == GATEWAY(c1,c2)
    } == 0
}
SLAVE :: else
```



RA: Aggergation

- Platziere Aggregatoren nahe den Datenquellen

```
AGG(0) :: { sensor == temp }
AGG(N) :: {
    count(2 hops) {
          role == AGG(N-1) &&
          dist(pos, sink_pos) >
              dist(super.pos, sink_pos)
    } >= 2 &&
    count(2 hops) {
          role == AGG(N)
    } == 0 &&
    1 <= N && N <= 2
}
```



RA: Eigenschaften

- Abstraktionsniveau
  - Dynamische Rollenzuteilung
  - Basisdienst, nur Teil einer Applikation
- Mächtigkeit
  - Grundlegende Abstraktion in vielen WSN Applikationen
- Effizienz
  - Lokalität von der Spezifikation abhängig

Zusammenfassung

- Programmierung von Sensornetzen komplexes Problem
  - Aufgabe: hohes Abstraktionsniveau, problemorientiert
  - Sensornetz: niedriges Abstraktionsniveau, systemorientiert, Verteiltheit
- Erste Ansätze versuchen Lücke zu verkleinern
- Gegenläufige Ziele
  - Hohes Abstraktionsniveau
  - Mächtigkeit
  - Effizienz (Energie, Skalierbarkeit)

Hinweis

- KuVS Fachgespräch „Systemsoftware für Pervasive Computing“
- 14-15 Oktober, Uni Stuttgart
- Abstract bis 19 September

www.nexus.uni-stuttgart.de/de/aktuelles/ereignisse/SystemSoftware2004.html