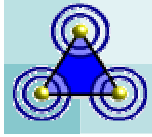
Middleware für mobile spontan vernetzte Sensornetzwerke

Jan Blumenthal, Dirk Timmermann
Universität Rostock

DFG-SPP 1140 Jahreskolloquium:
Basissoftware für selbstorganisierende Infrastrukturen für vernetzte mobile Systeme
24.-26.01.2005, Dagstuhl



MD

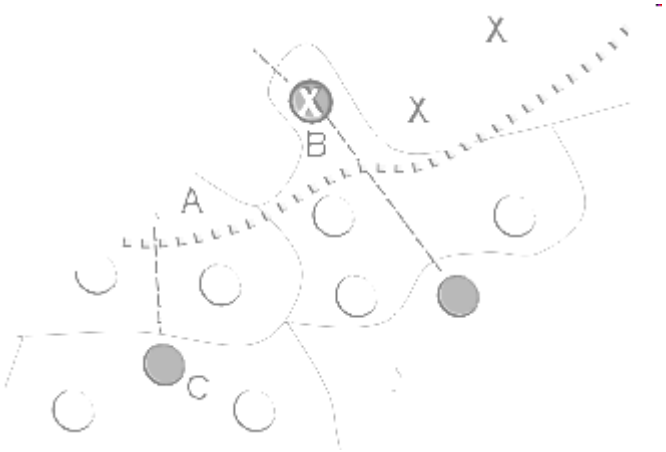
Übersicht

Ergebnisse

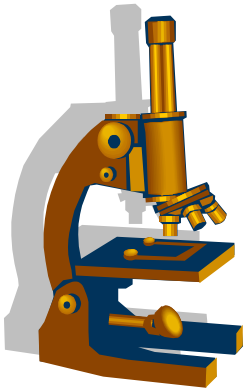
- SeNeTs
- Data Collection Protocol
- Coarse Grained Localization
- Weighted Centroid Localization

Aktuelle Forschung

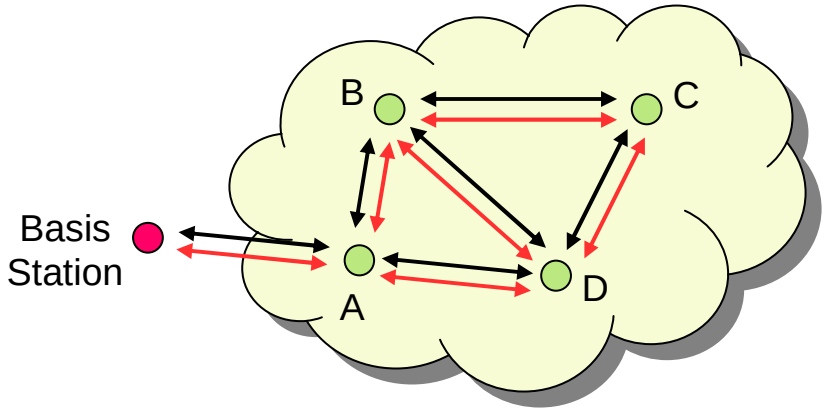
- EnviSense
- Mobile Positionierung
- Veröffentlichungen



– Ergebnisse –
SeNeTs



Test von Sensornetzwerksoftware



Drahtloser Kommunikationskanal

Kommunikation verursacht durch Anwendung

Kommunikation verursacht durch Tests und Administrierung

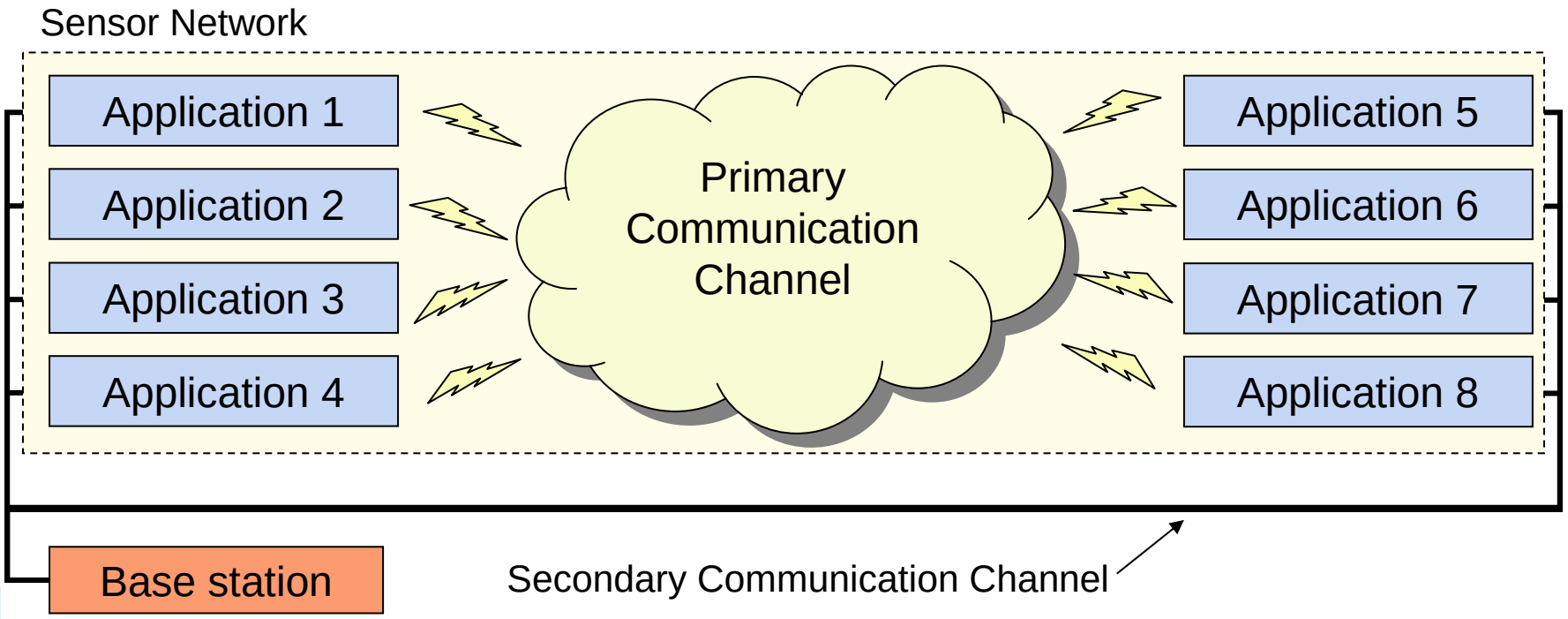
- Erhöhte Kommunikation
- Drahtlose Kommunikationskanäle agieren als **Flaschenhals**
- Resultate werden unrealistisch oder falsch

Ansatz

- Ausführen von Sensorknotenanwendungen auf PC‘s bestückt mit Bluetooth-Sensoren
- Anwendungen kommunizieren per Funk
- Administrierung via LAN-Backbone

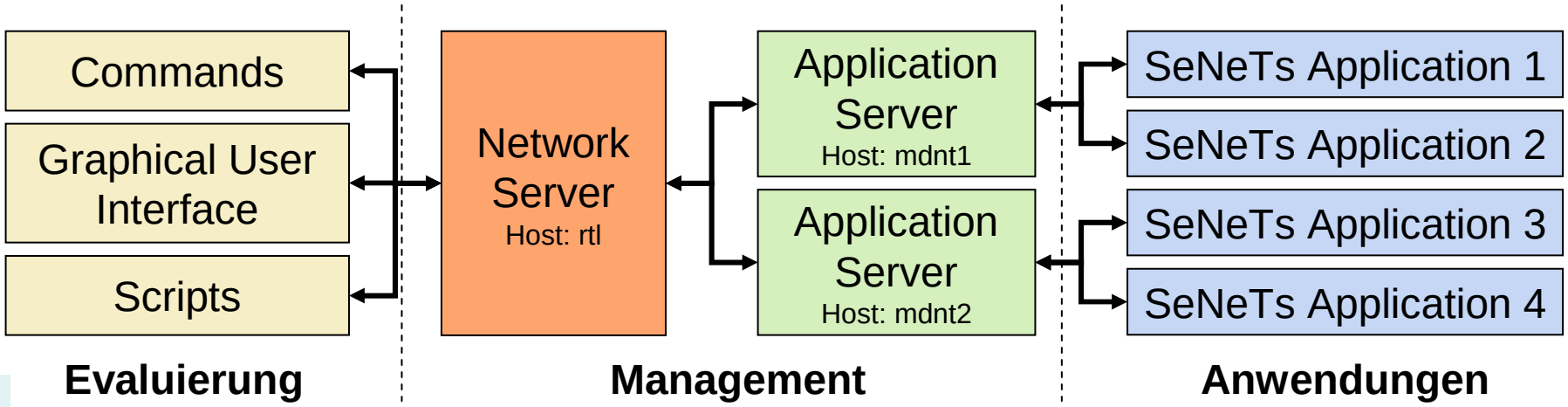
Kommunikationskanäle

- Anwendungen kommunizieren durch primären Kanal
- Administrierung durch sekundären Kommunikationskanal
  - … verhindert Flaschenhals-Effekt
  - … beseitigt Einfluss von Test und Administrierung auf den primären drahtlosen Kommunikationskanal



SeNeTs-Architektur

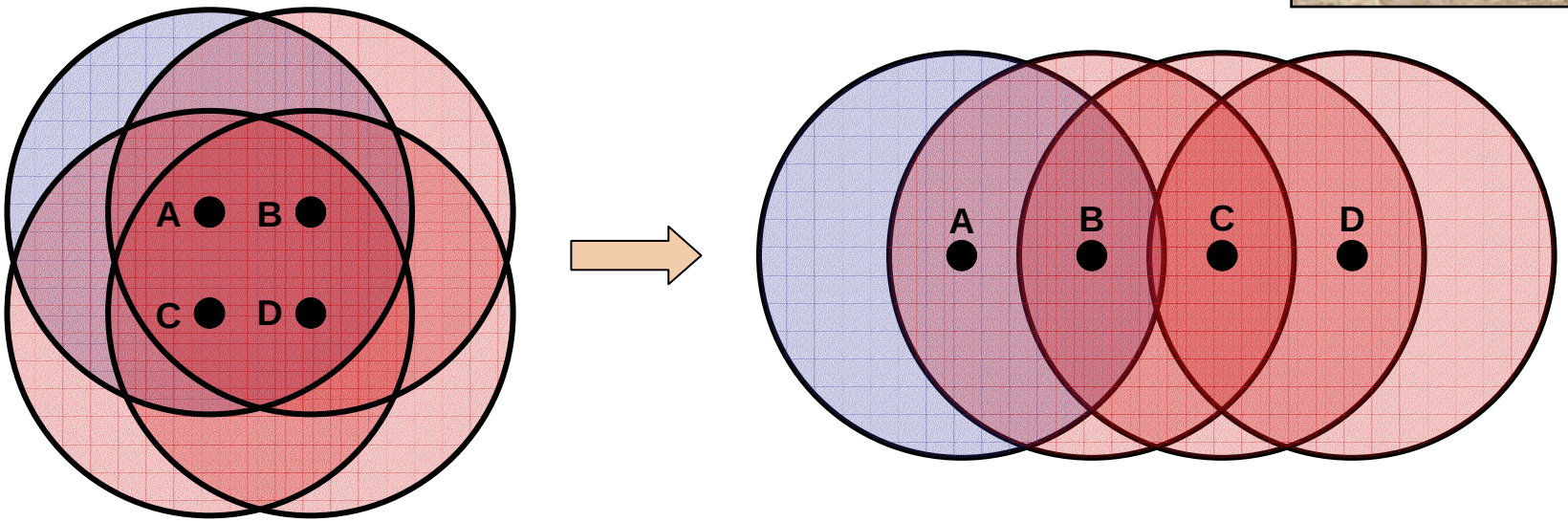
- Hierarchische Netzwerkarchitektur
- Loser Verbund der Komponenten
- Zusätzliche Features:
  - Logging und Steuerung von entfernten Sensorknoten
  - Software-Updates
  - Kommunikations-Management (sekundärer Kanal)



Positionsbasierte Filterung

- Physikalische und virtuelle Positionen stimmen nicht überein
- Alle Knoten innerhalb der gegenseitigen Übertragungsreichweite

Filterung aller Nachrichten, die von Knoten empfangen wurden, die sich nicht in virtueller Übertragungsreichweite befinden.



Physikalische Anordnung (Labor)

Virtuelle Anordnung (Deich)

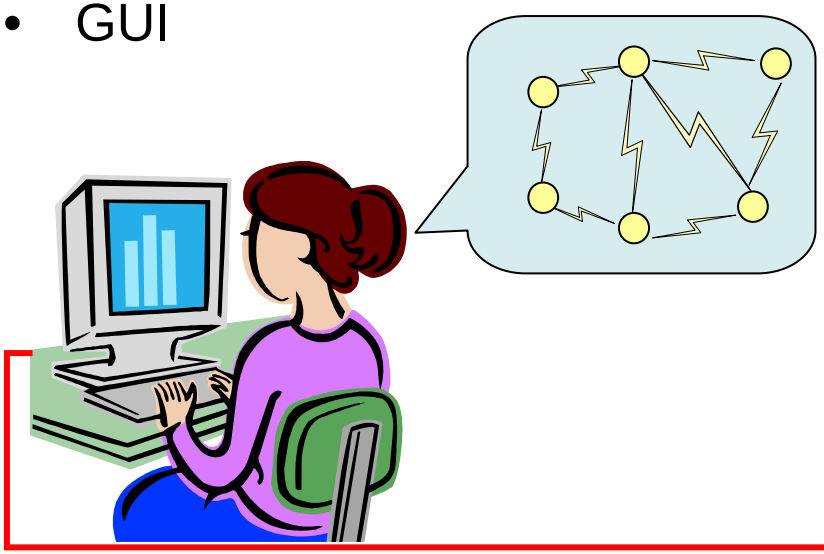
SeNeTs – Derzeitiger Stand

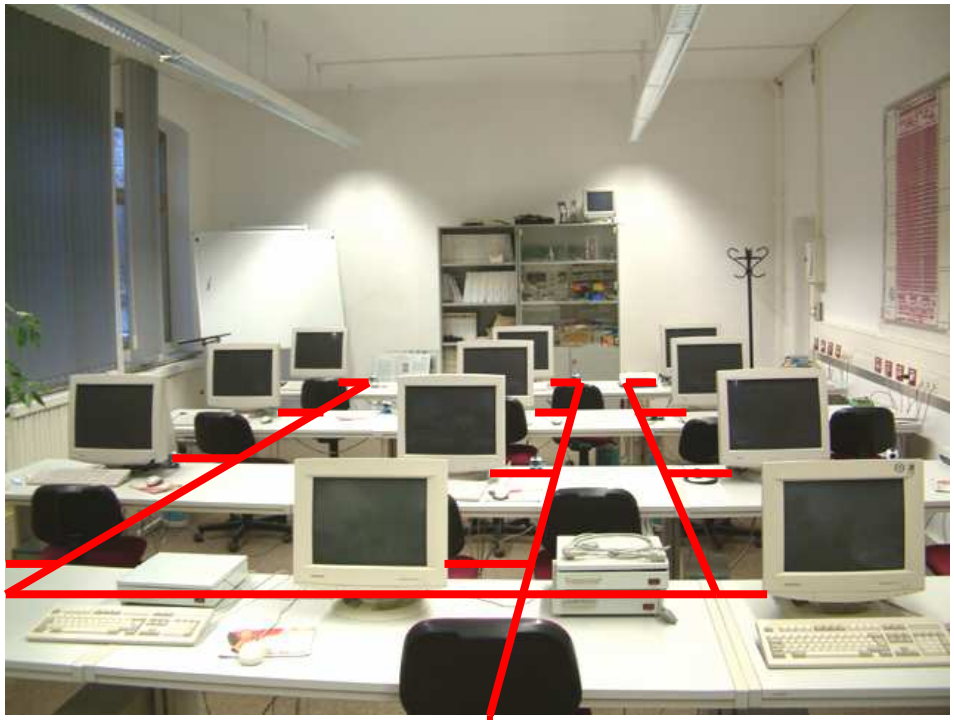
Bereits fertiggestellt:

- Plattformunabhängig (Linux, Windows)
- Zwei Beispiel-Anwendungen (ANSI C)
- Application-Server (C++)
- Network-Server (C++)

In Arbeit:

- Umgebungs-Management
- GUI



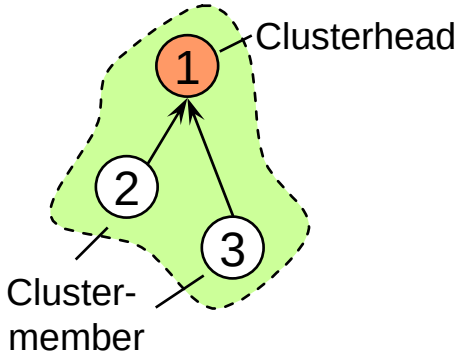


Commands via TCP/IP

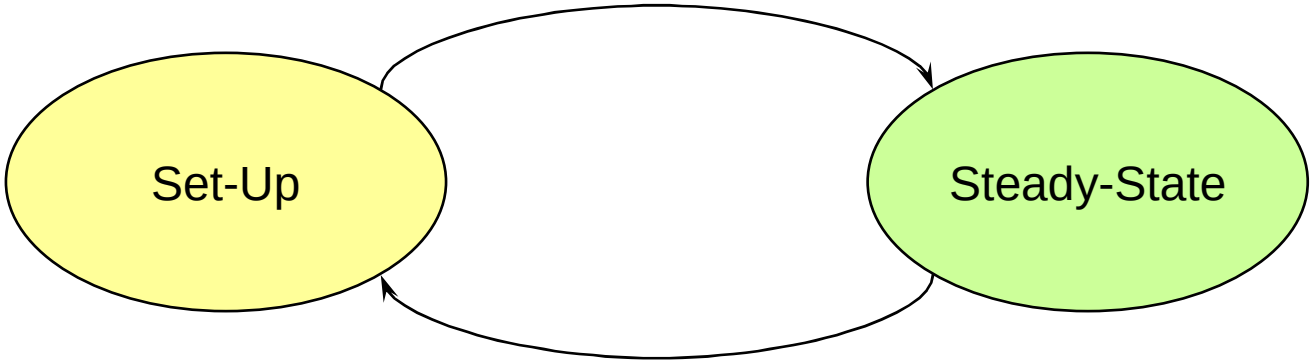
– Ergebnisse –
Data Collection Protocol

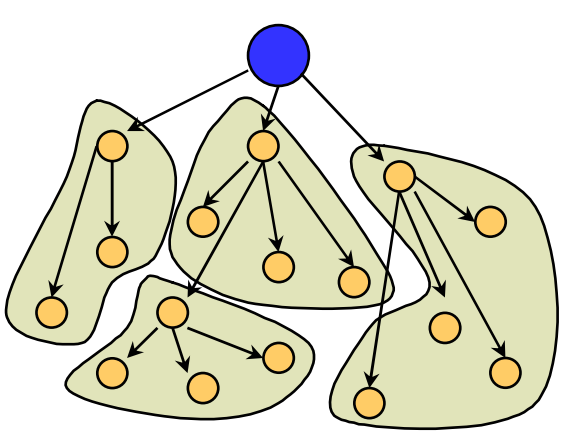
Data Collection Protocol (DCP)

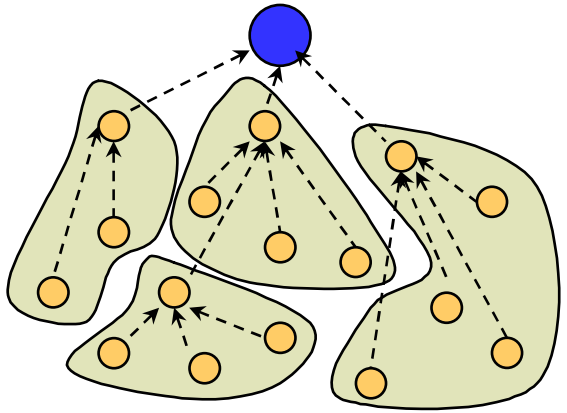
- Begriff: „Data Collection“
- Kooperationsstrategie: Clustering
  - Clusterhead, Clustermember
- Periodische Cluster Reorganisation
  - Gleichmäßiger Energieverbrauch
  - Einbeziehen von Topologieänderungen
- Bluetooth
- DCP baut Clusterhierarchien, enthält aber kein Verbindungsmanagement (im Gegensatz zu Scatternetzen!)



Data Collection Protocol – 2 Phasen



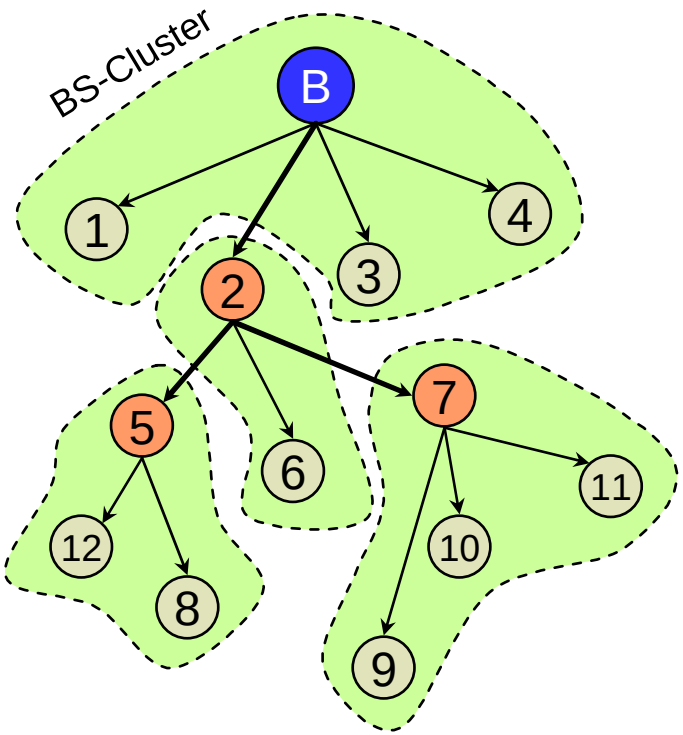




- Wahl des Clusterheads
- Cluster-Formierung
- Aufbau des Routings

Sammeln der Sensordaten

DCP – Set-Up-Phase (I)

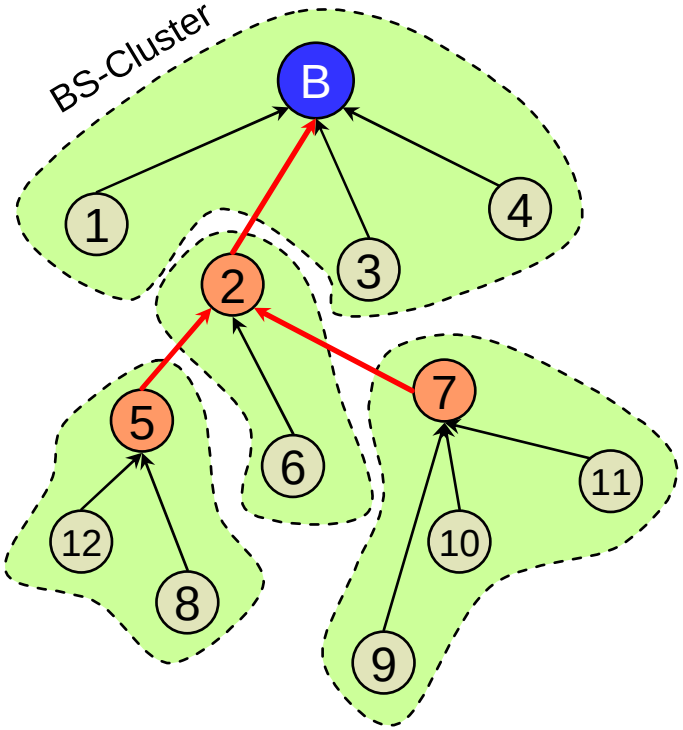


1. Wahl des Clusterheads
   - zufällig bestimmt
2. Base Station Inquiry
   - Ermitteln des CH und des 1-hop-CM
3. Basisstation überträgt PFA
   - erst CMs (1,3,4), dann CHs (2)
4. Entdeckte Knoten schalten Inquiry Scan ab
   - „invisible“ mode
5. 1-hop entferntes CH inquiry
   - bereits entdeckte CM and CH werden durch BS nicht mehr detektiert, da Inq.-Scan aus
6. 1-hop entfernte CH übertragen PFA
   - erst CMs (6), dann CHs (5,7)

Abkürzungen:

BS	= Basisstation
CM	= Clustermember
CH	= Clusterhead
PFA	= Packet Forward Address

DCP – Steady-State-Phase



- Clustermember überträgt Sensordaten zum Clusterhead
- CH verarbeitet Sensordaten vor (Data Compression/Fusion)
- Clusterhead sendet aggregierte Daten an PFA/Basisstation

Abkürzungen:

BS = Basisstation
CM = Clustermember
CH = Clusterhead
PFA = Packet Forward Address

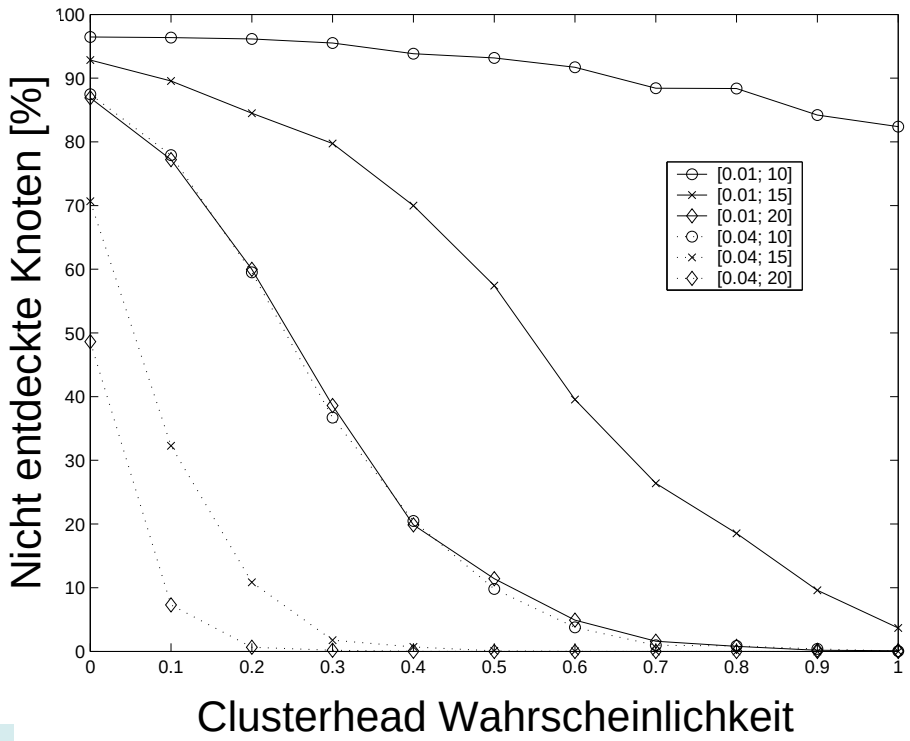
Knoten trennen Verbindung sofort nach Datenübertragung (im Gegensatz zu Scatternetzen)

- Cluster sind nicht begrenzt durch Piconetzgröße
- Energieeinsparungen bei geringen Datenraten
- Reduzierte Interferenzen (Wieviele Bluetooth Piconetze passen in einen Raum?)

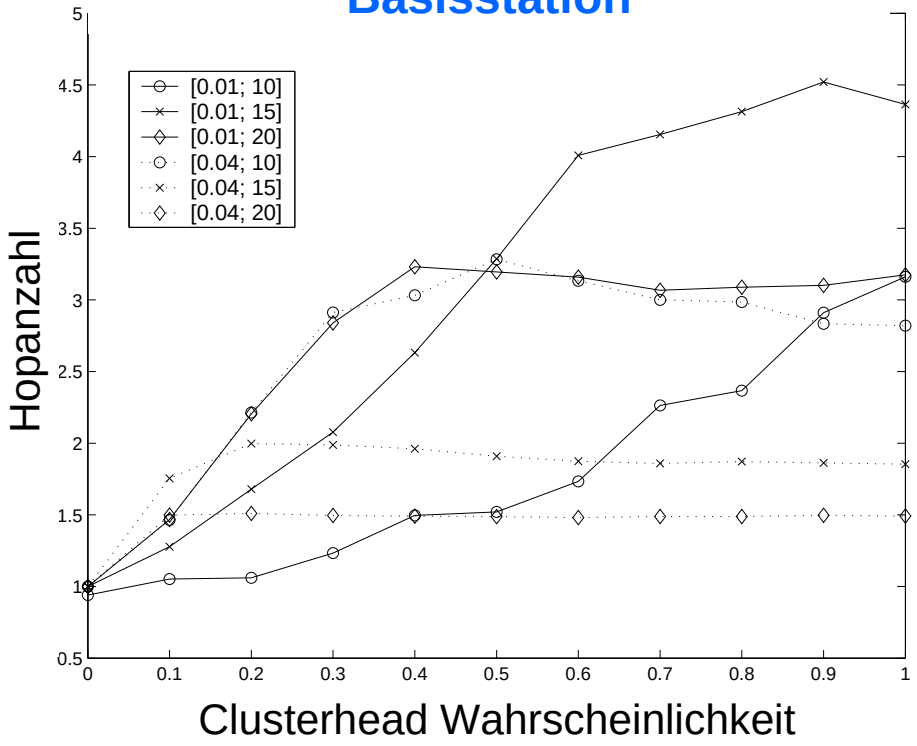
Simulationsresultate

Legende:	[Knotendichte; Übertragungsreichweite]
2 Simulation-Setups:	100m*100m mit 100 Knoten (Knotendichte 0,01)
	50m*50m mit 100 Knoten (Knotendichte 0,04)

Nicht entdeckte Knoten

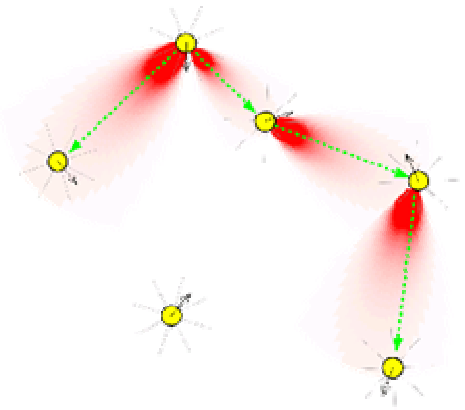


Durchschnittliche Hopanzahl zur Basisstation

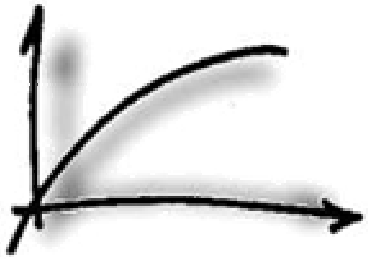


DCP-Entwicklung mit SeNeTs

- Extensive Tests und Emulationen von DCP-basierenden Netzwerken
- Beweis des Konzeptes
- SeNeTs-Emulation bietet essentielle Einsicht in das Protokoll-Design
  - Synchronisierung von Page/Inq. Scan-Modi
  - Test der Multihop-Funktionalität
  - Beobachtung des tatsächlich anfallenden Netzwerkverkehrs



– Ergebnisse –
Positionierung CGLCD



Coarse Grained Localization

Eigenschaften:

- Schwerpunktbestimmung aus Beaconpositionen
- Fehler=Distanz zwischen bestimmter Position und wahrer Position

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

$x_{i_{app}}, y_{i_{app}}$ = Bestimmte Koordinaten von Knoten i

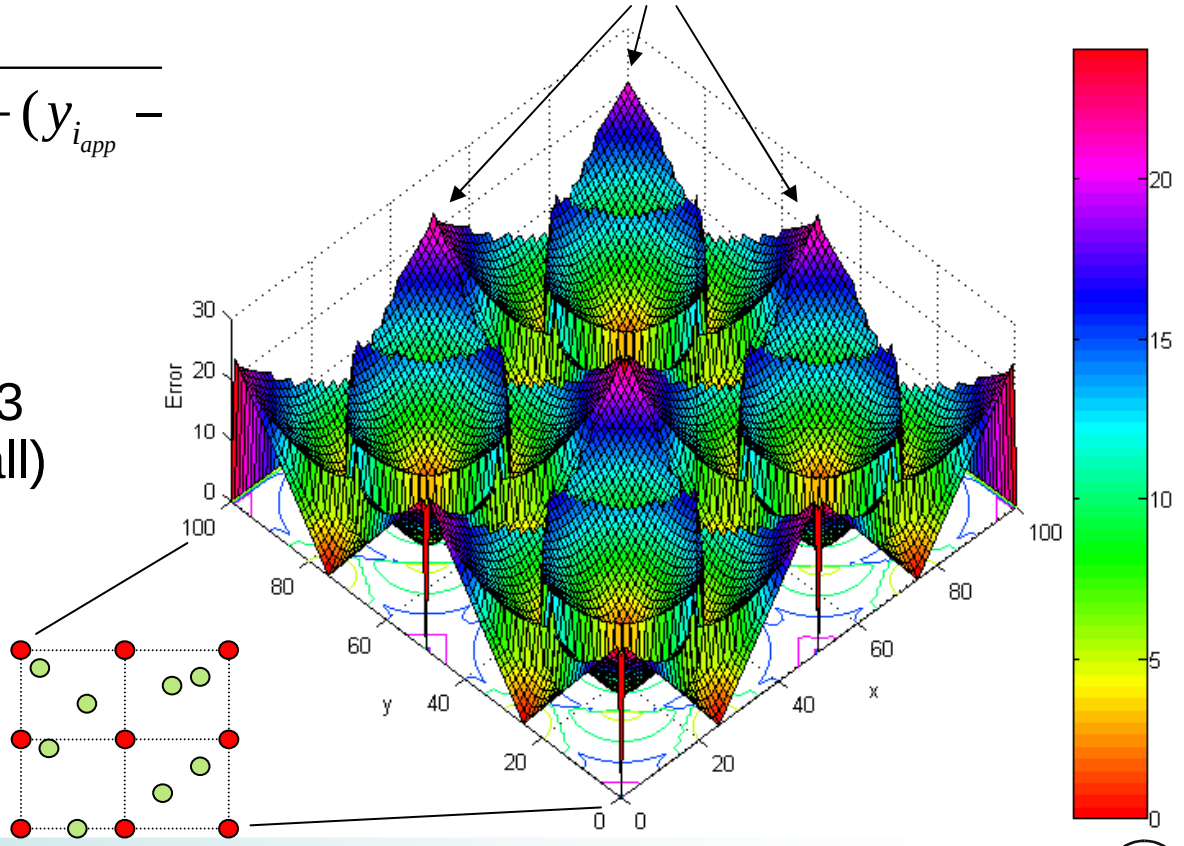
x_{i_a}, y_{i_a} = Exakte Koordinaten von Knoten i

f_i = Positionierungsfehler von Knoten i

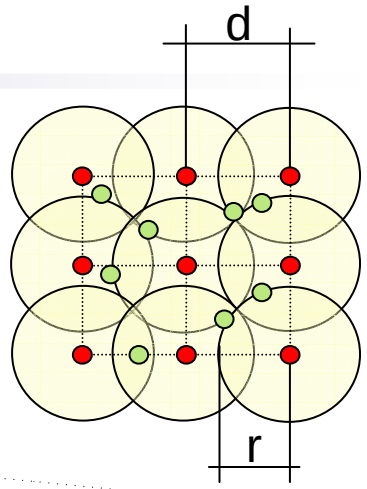
= Beacons

Fehlerverlauf:

- Gitteranordnung von 3x3 Beacons (Infrastrukturfall)
- Feldbreite 100x100
- Übertragungsreichweite der Beacons r=50



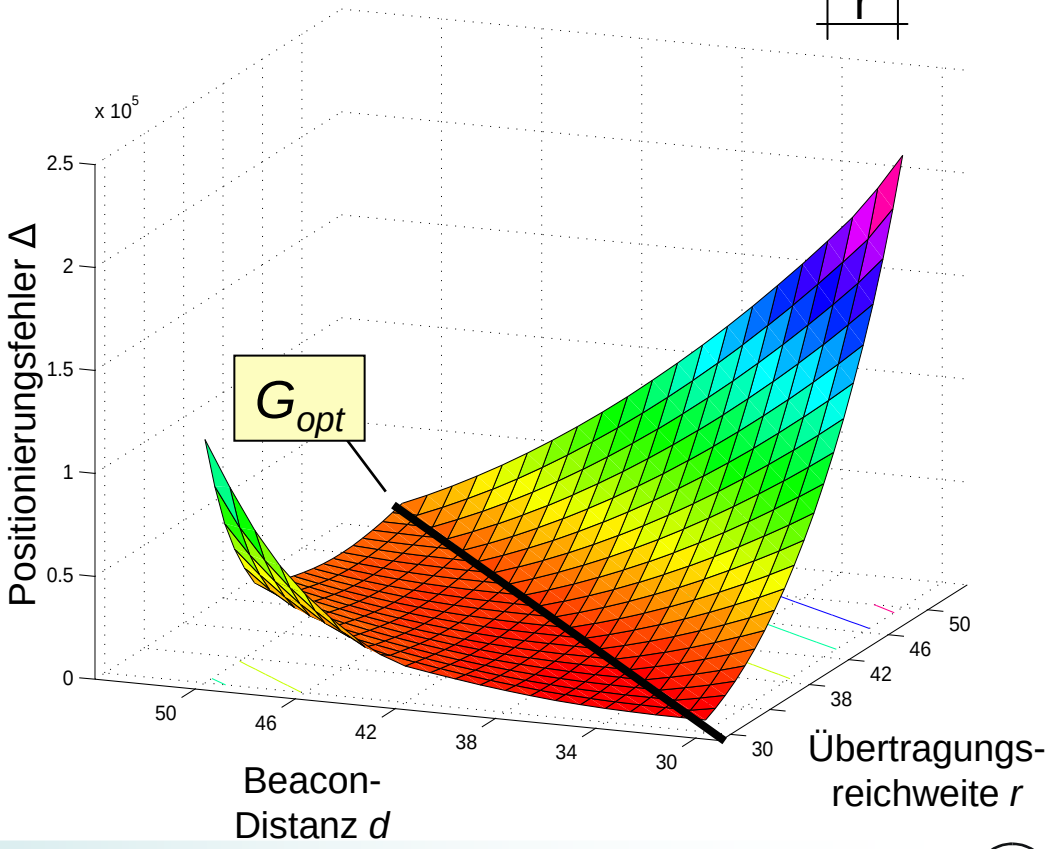
Bestimmung des minimalen Fehlers



Gesucht: Optimales Verhältnis G_{Opt} aus Übertragungsreichweite vs. Beacondistanz

- Bestimmung des Positionierungsfehlers Δ_i durch Variation von *r* und *d*
- Bestimmung von r_{opt} an Distanzen d_i mit kleinstem Positionierungsfehler

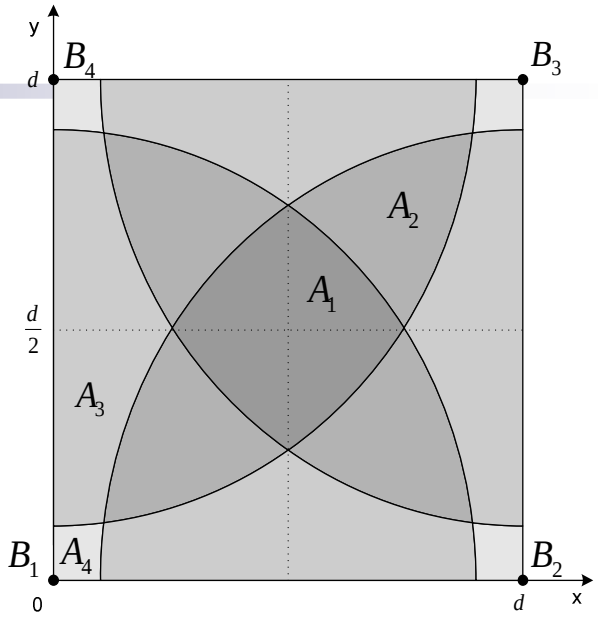
$$G_{opt_i} = \frac{r_{opt_i}}{d_i}$$



Ergebnis:

- G_{opt} is konstant !
- $G_{opt} \approx 0{,}86$

Beweis der Optimalität



Beweis:

- Ermittlung des minimalen Flächen-fehlers P_x eines Infrastrukturelements
- r_{opt} aus min. Fehler $P_{1..i}$ berechnen

Ergebnisse:

- Beweis von C_{opt}/r_{opt} =0,86 entwickelt
- Komplett analytische Beschreibung des Algorithmus (Keine Simulation nötig!)
- Minimaler Fehler: ~5%
- Reduzierung des Energieverbrauchs

$$P_4(r,d) = 2\left|\int_0^{\frac{d}{2}-\frac{\sqrt{2r^2-d^2}}{2}} \int_x^{d-\sqrt{r^2-x^2}} \sqrt{x^2+y^2}\,dy\,dx\right|$$

$$\left.\frac{dP_A(r,d)}{dr}\right|_{d=const} = 0 \rightarrow r_{opt}$$

$$E_{Trans} = E_{Bit}\cdot\left(\frac{4\cdot\pi\cdot r_{opt}}{\lambda}\right)^2 < E_{Bit}\cdot\left(\frac{4\cdot\pi\cdot r_{\max}}{\lambda}\right)^2$$

r = Übertragungsreichweite
d = Abstand zwischen Beacons
P_x = Flächenfehler für A_x
E = Energie

Power-Error Produkt (PEP)

Energiebetrachtungen (ideal):

$$E_{Send} = E_{Init} + mE_{Dyn}$$

$$E_{Dyn} = E_{Bit} \cdot \left(\frac{4 \cdot \pi \cdot r}{\lambda} \right)^2$$

$$E_{Dyn}\Big|_{E_{Bit}\cdot\left(\frac{4\cdot\pi}{\lambda}\right)^2=1} = r^2$$

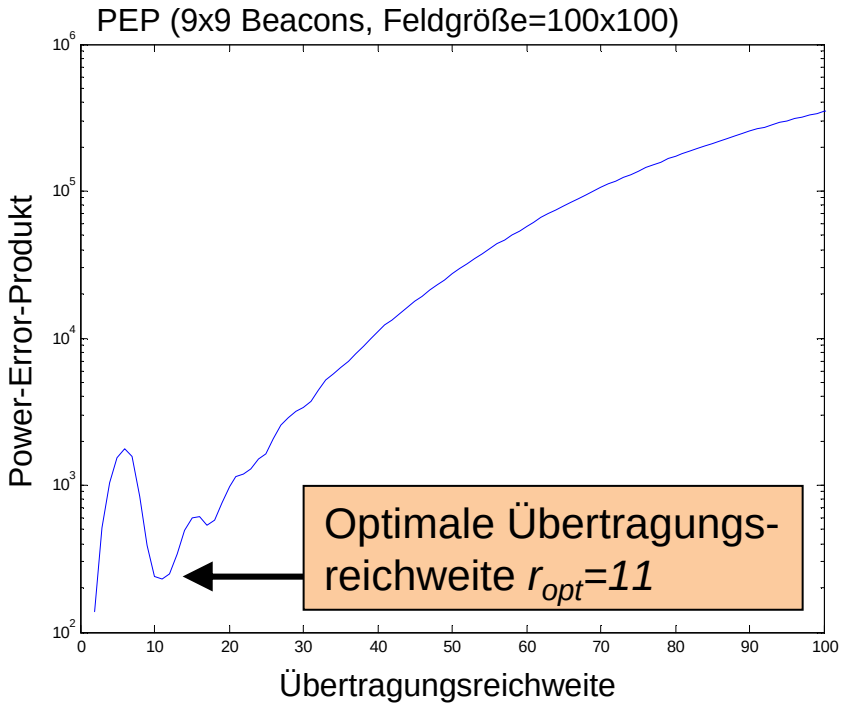
m	= Anzahl der zu übertragenden Bits
E_{Init}	= Initialisierungsenergie des Transmitters
E_{Dyn}	= Übertragungsenergie für ein Bit
λ	= Wellenlänge
r	= Übertragungsreichweite der Beacons
PEP	= Power-Error-Produkt
L	= Anzahl der Sensorknoten im Netzwerk
r	= Distanz
f_i	= Positionierungsfehler in einem Sensorknoten
$f_{l_{mean}}$	= Durchschnittlicher Positionierungsfehler

Power-Error-Produkt:

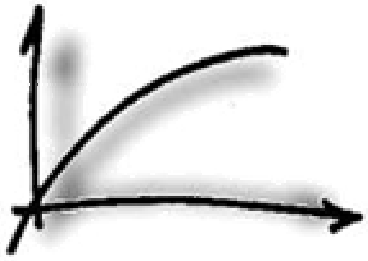
$$PEP = E_{Dyn} \cdot f_{l_{mean}}$$

$$= r^2 \cdot \frac{\sum_{i=1}^{l} f_i}{l}$$

Performance-Indikator für Optimierungs-problem aus Übertragungsreichweite, Unbekannten und Positionierungsfehler



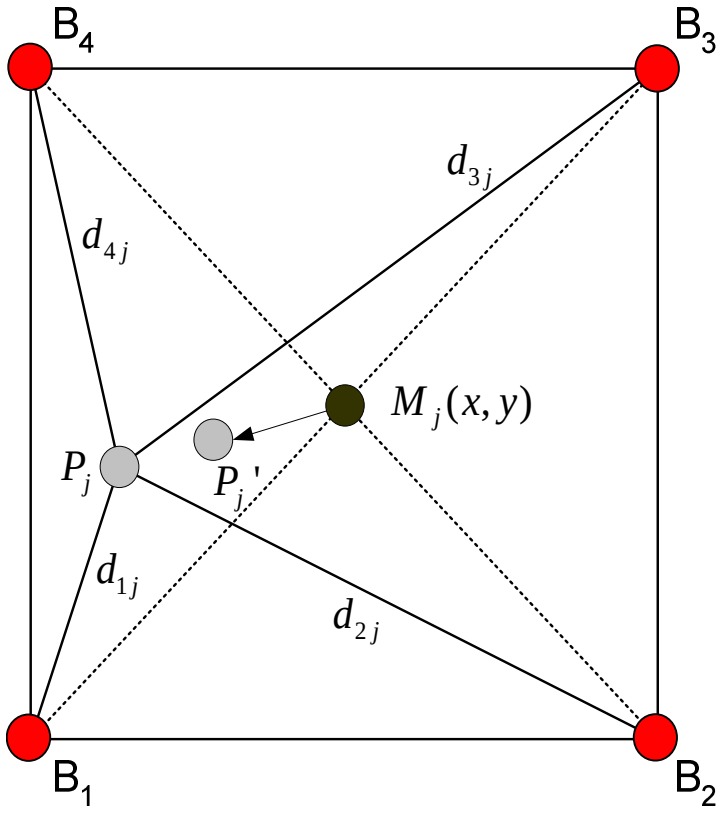
– Ergebnisse –
Positionierung WCL



Weighted Centroid Localization (WCL)

Ansatz: Gewichtete Schwerpunktberechnung

$$P_j'(x,y) = \frac{\left(\sum_{i=1}^{b}\left(w_{ij}\cdot B_i(x,y)\right)\right)}{\left(\sum_{i=1}^{b} w_{ij}\right)}$$



Optimierungskriterium:

$$w_{ij} = \frac{1}{\left(d_{ij}\right)^{g}}$$

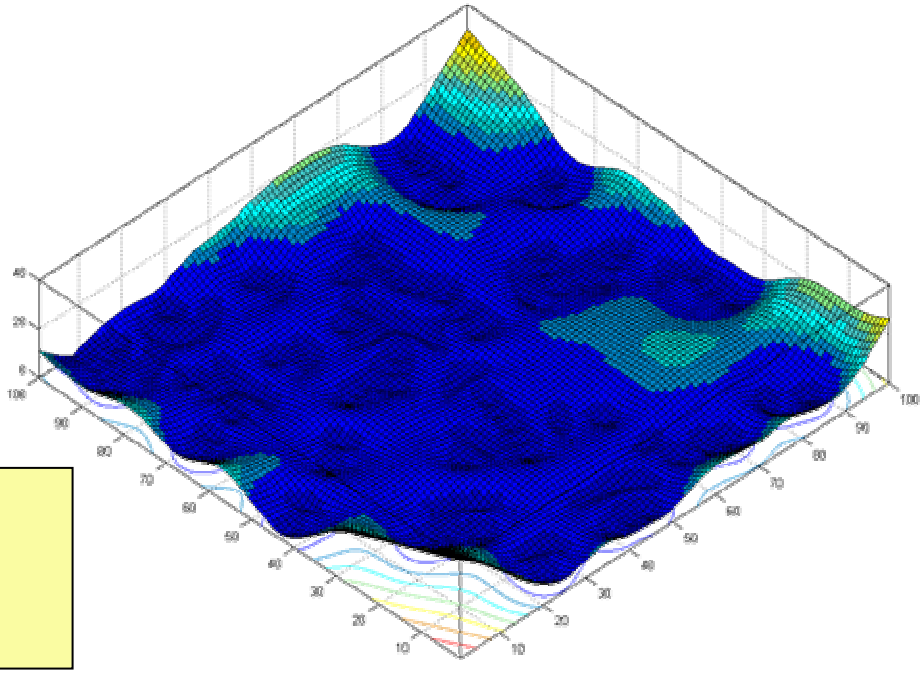
w_{ij}=Gewicht zwischen B_i und U_j; b=Beaconanzahl; $B_j(x,y)$=Position des Beacons

WCL - Eigenschaften

- Zufällige Verteilung der Knoten und Beacons
- Autarke Berechnung
- Robust & skalierbar
- Geringerer Energieverbrauch durch

  ↓ Berechnungsaufwand

  ↓ Netzwerkverkehr
- Positionierungsfehler gering:
  - WCL.....>3%
  - APIT.....>6%
  - CGLCD..>5%



Beispiel:

1000 Knoten bei 25 Beacons (2,5%)

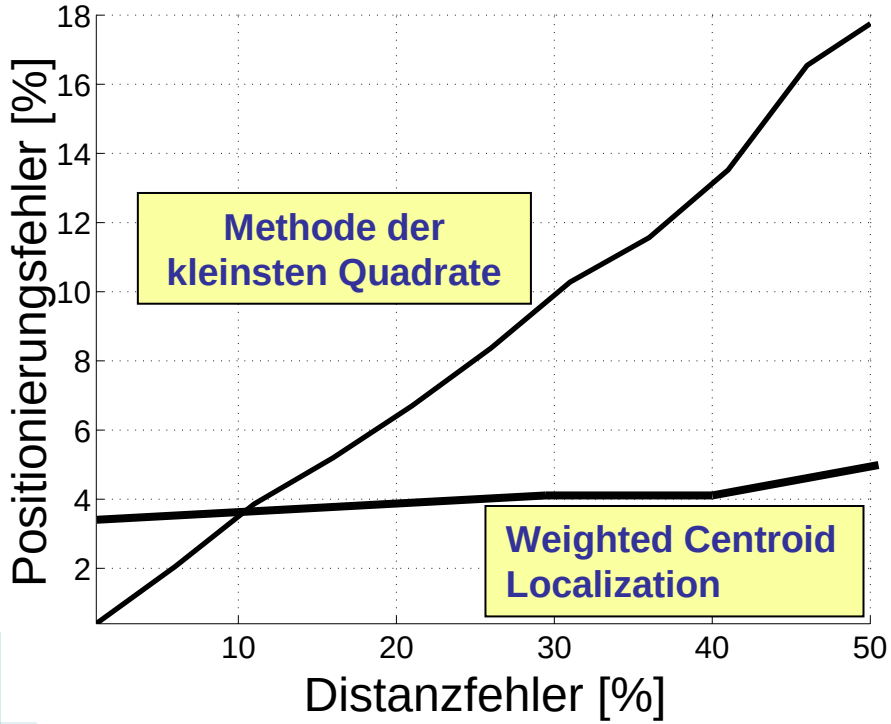
Ø Positionierungsfehler=3,3%

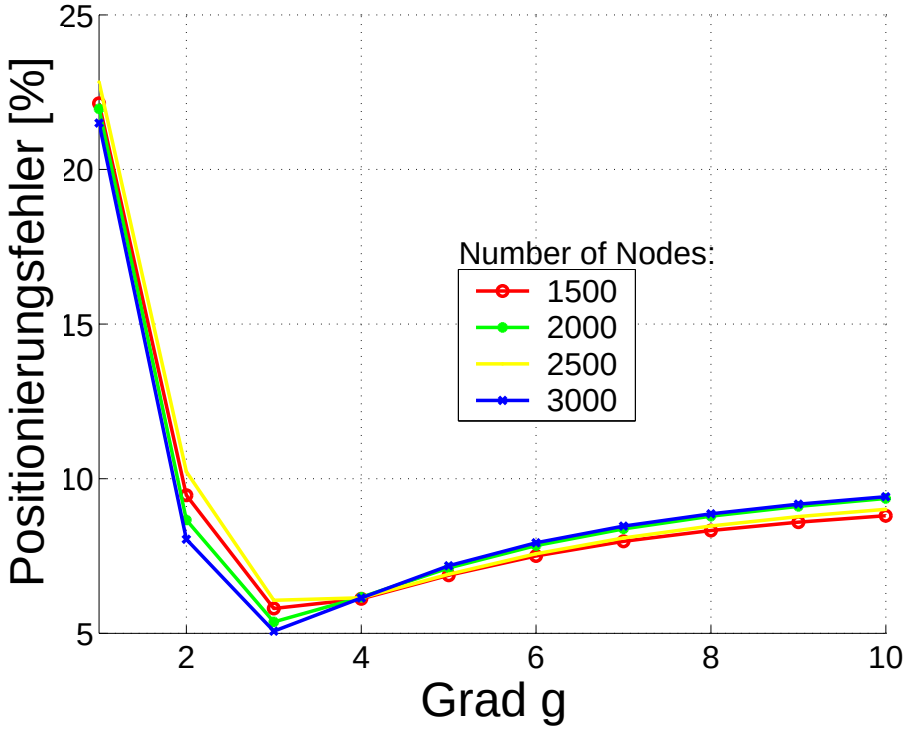
WCL - Besonderheiten

Unabhängigkeit von Eingangsmessfehlern steigt !

Positionierungsfehler sinkt, wenn sich Grad des Gewichts 3 nähert.

$$w_{ij} = \frac{1}{\left(d_{ij}\right)^{g}}$$ ← Grad

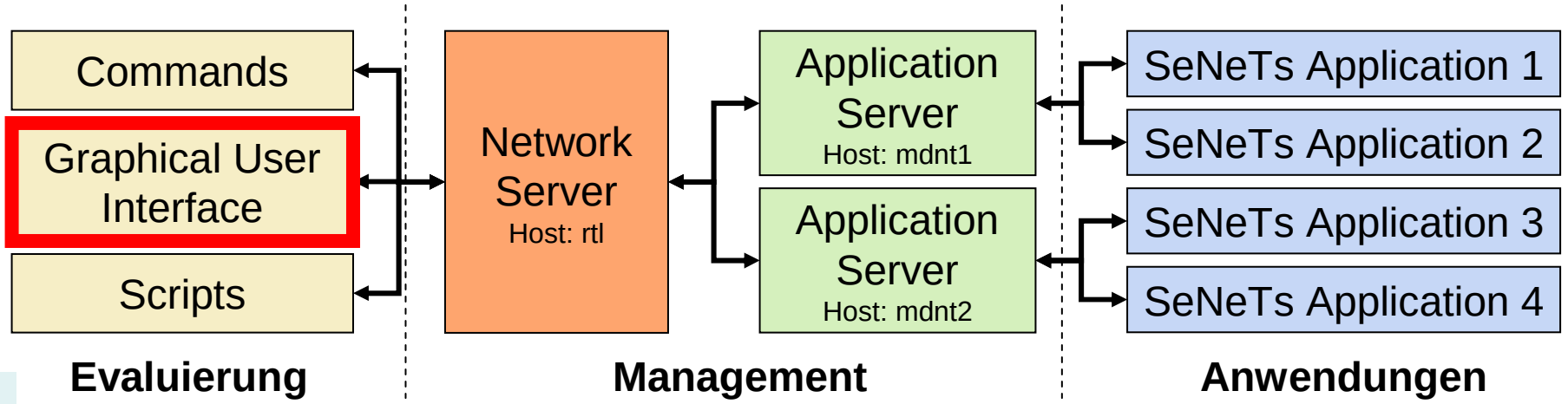




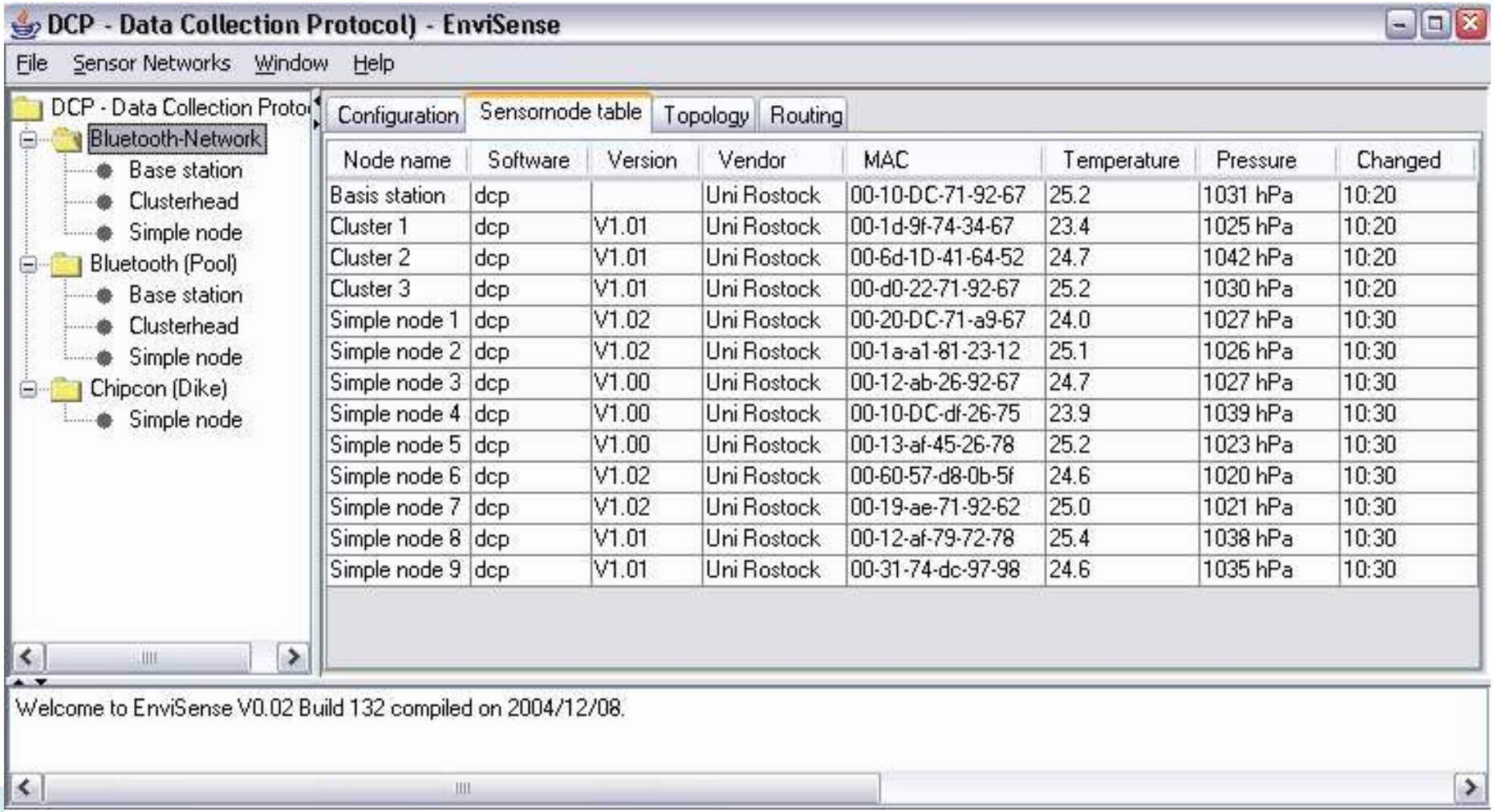
– Aktuelle Arbeiten –
EnviSense

EnviSense

- Leistungsstarke grafische Benutzeroberfläche für große drahtlose Sensornetzwerke
- Plattformunabhängig durch JAVA & XML
- Einfache Konfiguration von Sensorknoten (Start, Stop, Download, Reset, GetAttribute)
- Maßgeschneiderte Anfragen an Sensorknoten
- Übersichtliche Darstellung und Auswertung von Attributen der Sensorknoten

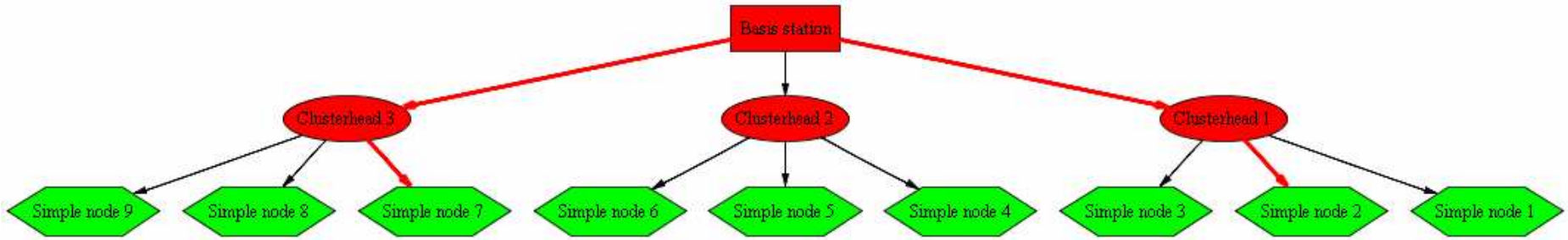


EnviSense: Screenshot

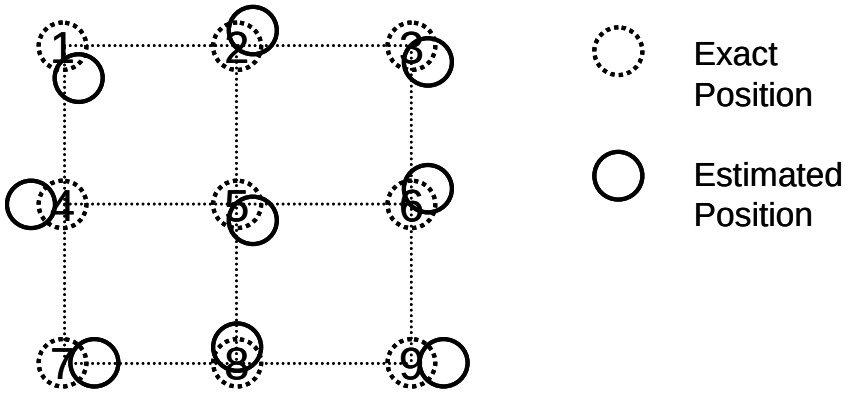


Auswertungen in EnviSense

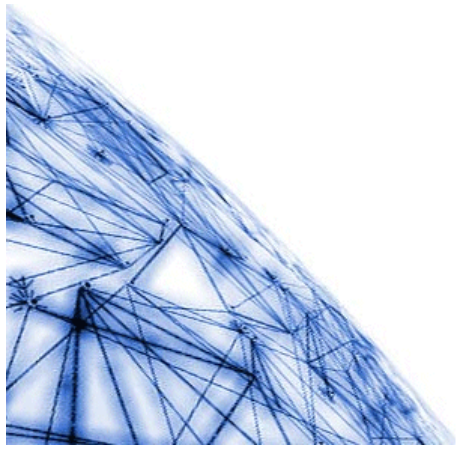
Topologiebaum



Positionierungsfehler

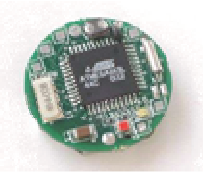


– Aktuelle Arbeiten –
Mobile Positionierung



Komplexität bei mobiler Positionierung

- ChipCon CC1010:
  - 8051 Mikrocontroller
  - Takt: Bis zu 16MHz → 16 Mips
- Atmel ATmega 128 (Mica Mote)
  - Takt: 4 MHz → 4 Mips



Beispiel: Berechnungsdauer auf Prozessor mit 10 Mips

n	O(n)	$O(n^2)$	$O(n^3)$	$O(n^4)$
10	10^{-6} s	10^{-5} s	10^{-4} s	10^{-3} s
100	10^{-5} s	10^{-3} s	10^{-1} s	10 s
1000	10^{-4} s	10^{-1} s	10^{2} s (≈ 1,7 min)	10^{5} s (≈ 1,2 Tage)
10000	10^{-3} s	10^{1} s	10^{5} s (≈ 1,2 Tage)	10^{9} s (≈ 31 Jahre)

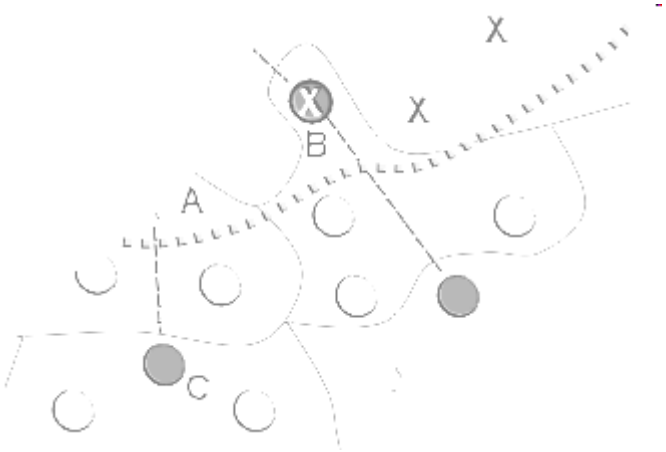
Komplexität bei Mobilität

Aber: Diese Prozessoren haben keine Floating Point Unit (FPU) !

Mobile Positionierung

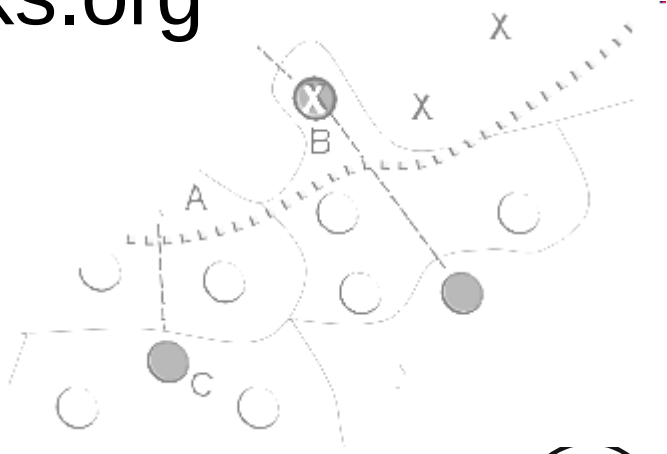
Exakte mathematische Verfahren kaum nutzbar, weil:

- Keine FPU vorhanden
- Komplexität der Algorithmen zu hoch
- Riesiger Speicherverbrauch durch Matrizen
- Hoher Energieverbrauch

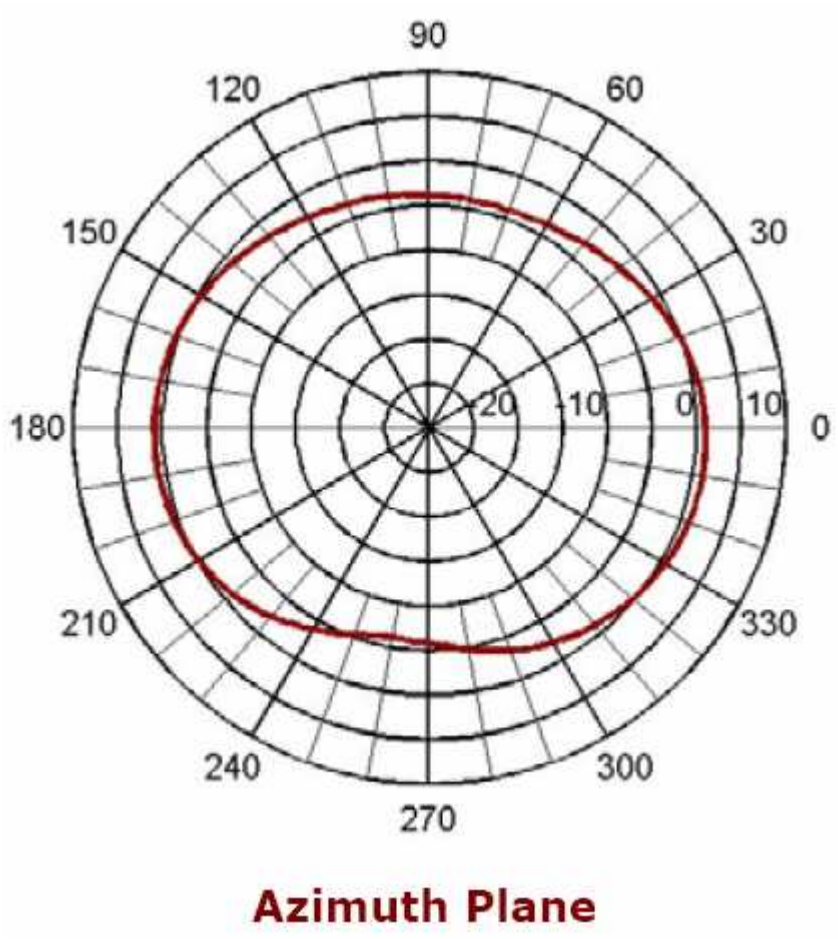


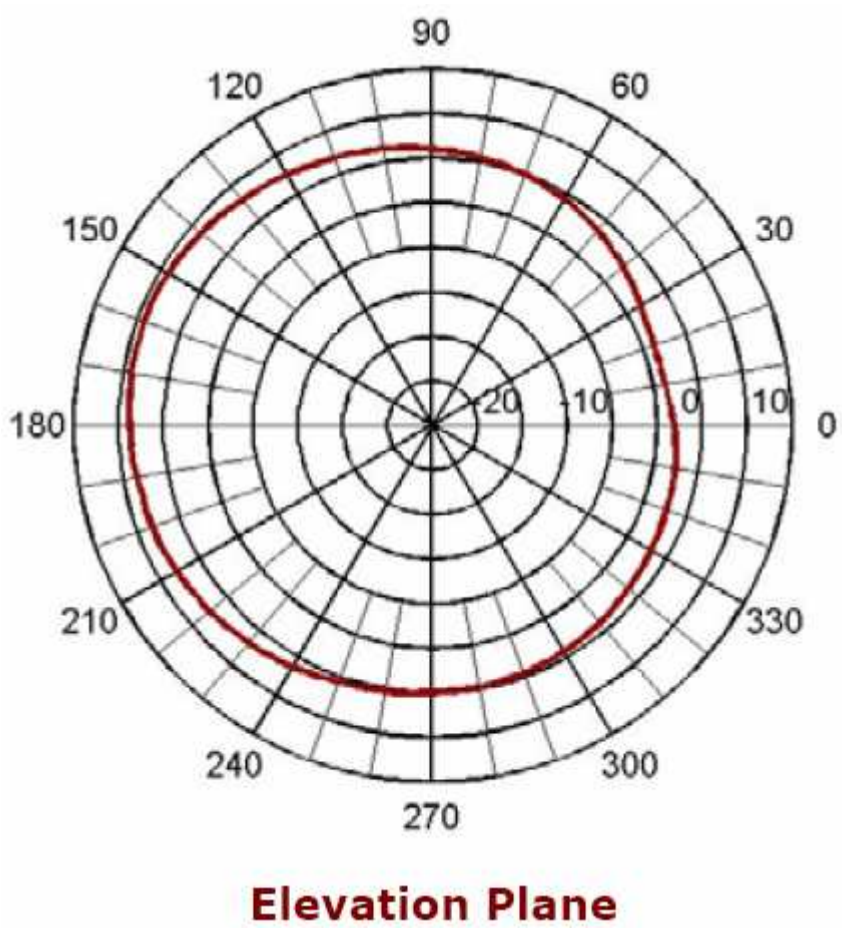
Vielen Dank!

www.sensornetworks.org



Empfangsprofile von Antennen





Quelle: http://www.centurion.com/home/antennaProd/sphere.asp

Demonstratorplattform: BlackCubes

Chipcon CC1010 – Development Kit

- Sehr geringe Leistungsaufnahme (9,1 mA in RX)
- Hoch sensitiv
- Einstellbare Ausgangsleistung (868 MHz)
- Datenraten bis zu 76,8 KBit/s
- Wenige externe Komponenten
- RSSI-Messungen
- Temperatursensor
- 8051-kompatibler Mikrocontroller