ETH Eidgenössische Technische Hochschule Zürich Swiss Federal Institute of Technology Zurich

inf | Informatik Computer Science

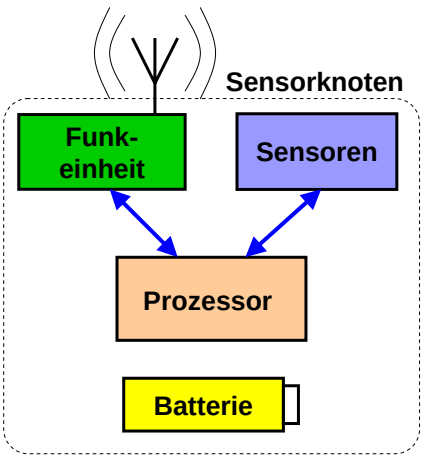
Building a Distributed System with Embedded Devices and Sensors

René Müller muellren@inf.ethz.ch
Prof. Gustavo Alonso alonso@inf.ethz.ch
Institut für Pervasive Computing

 8.12.2007

Drahtlose Sensornetze

- Ad-hoc Netzwerk aus Sensorknoten
  - 10/100/1‘000/10‘000 ???
- Def: Sensor-Knoten = ein Prozessor + eine Funkeinheit + einige Sensoren + eine Batterie
- Sensorknoten sind
  - Klein (wenige cm^3)
  - Billig (CHF 150 – …) ???



Sensorknoten

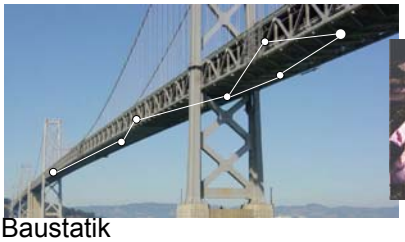
BTnode (ETH Zürich)
- CPU: ATmega, 8MHz
- Flash-Speicher: 128kB
- RAM: 4kB (intern) + 256kB (extern)
- Funk: CC1000 (76kbps) + Bluetooth

tmote sky (Moteiv)
- CPU: MSP430, 1MHz
- Flash-Speicher: 48kB
- RAM: 10kB (intern)
- Funk: CC2420 (250 kbps, IEEE 802.15.4)

Anwendungen

- **Überwachen/Messen/Kontrollieren** der Umgebung
  - Zivil/Militär
  - Naturwissenschaften (Biologie, Geologie, …)
  - Bau und Architektur (Intelligente Häuser, Brücken, etc.)

Landwirtschaft



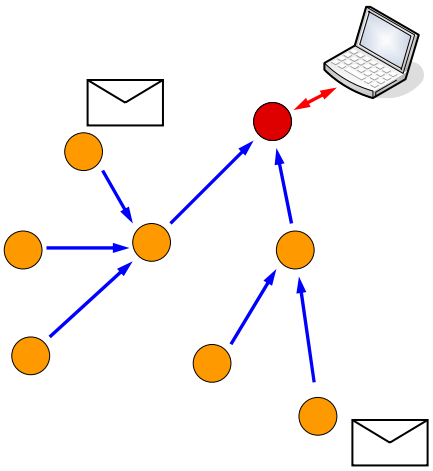
Baustatik

Schweinemast

Seismik

Häufiger Anwendungsfall

- Sensorkonten, die phys. Grösse X messen, werden verteilt
- Gemessene Daten werden auf PC/PDA aufgezeichnet od. weiter verschickt.
- PC/PDA/… via Sensor mit Sensornetz verbinden
- Knoten bauen Ad-hoc Netzwerk auf (z.B., Baum)
- Knoten senden periodisch Messungen an PC/PDA.



Inhalt

- Sensornetze
  - Herausforderungen
  - Aspekte aus “Verteilte Systeme”
- Routing-Bäume
- Aggregation von Messwerten im Netzwerk
- SwissQM Demo
  - Virtuelle Maschine für Sensornetze
  - Verarbeitung von kontinuierlichen Queries

Herausforderungen

- Ressourcen-Knappheit
  - Energie/Batterie (J)
  - Speicher (kB) / Rechenleistung (MIPS)
  - Kommunikations-Bandbreite (bit/s)
- Fehlertoleranz und Robustheit
  - Ausfall einzelner Knoten
- Skalierbarkeit
  - Netzwerkgrössen: 10, 100, 1‘000, …

Problem #1: Energie

- Sensorknoten sind batteriebetrieben
- Austausch der Batterien ist meist nicht praktikabel.
- Energie für Sensorknoten: aus 2x AA Alkali-Batterien
  E ≈ 17 kJ (= 4 kcal)

  [Duracell PLUS, 250 mA Entladestrom bis 0.9 V Klemmspannung]

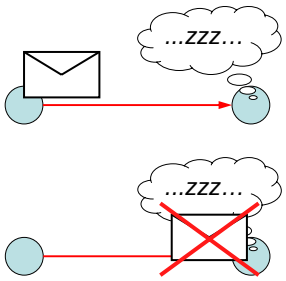
Vergleich
Energieinhalt eines Hamburgers

© McDonands

E ≈ 3556 kJ (= 850 kcal)
[Quelle: BigMac, McDonalds]

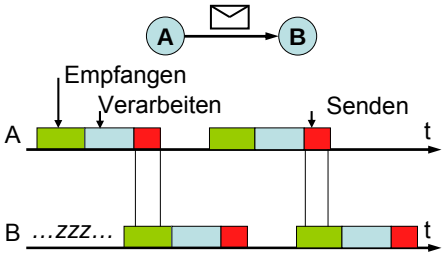
Energie Management

- Verbraucher (Bsp. tmote sky):
  - Prozessor (≈0.8-3 mA)
  - Sensoren (≈1 mA)
  - Funkeinheit (Senden oder Empfänger an ≈ 20mA)
- **Energiesparen**: zur Zeit nicht benötigte Verbraucher ausschalten
- **Tastgrad** (duty cycle) = Dauer "An"-Phase / Gesamtdauer
  - 100% immer "An"
- Grösster Verbraucher ist Funkempfänger! ← optimieren



- An/Aus-Phasen der Empfänger müssen synchronisiert sein
  - Logische Zeit (Lamport) alleine reicht hier nicht!
  - Synchronisation der Uhren nicht der Zeitstempel von Nachrichten

Synchronisation der Knoten



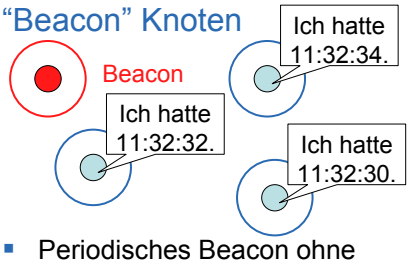
- Sende-Phase und Empfangs-Phase von Sender und Empfänger müssen überlappen.
- Wann beginnt/endet eine Phase? Koordination?

- Auch An/Aus-Phasen innerhalb des Netzwerk-Stacks muss koordiniert sein.
  - Applikation sendet periodisch Messungen
  - Routing Layer sendet Routing Tabelle an Nachbarn (periodisch oder auf Anfrage)
  - MAC Layer: Collision Avoidance, Random Back-off, etc.
- Wann die Funkeinheit ein/ausschalten? Wer darf das?

→ Cross-Layer Optimierungen

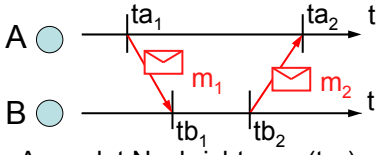
Uhren-Synchronisation

"Beacon" Knoten



- Periodisches Beacon ohne Zeitangabe dient zur Synchronisation.
- Nach Empfang des Beacons sendet jeder seine lokale Zeit.
- Knoten passen Schedule an, d.h. wann Nachbar "wach" ist eine Meldung empfangen kann.

Paarweise Sychronisation

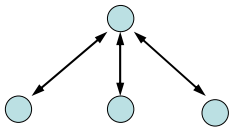


- A sendet Nachricht m_1= (ta_1) mit Zeitstempel ta_1.
- B empfägt m_1 zu Zeit tb_1.
- B sendet m_2=(ta_1,tb_1,tb_2).
- A empfängt m_2 zu ta_2.
- A berechnet relativen Drift Δ und mittlere Laufzeit d

$$\Delta = \frac{(tb_1 - ta_1) - (ta_2 - tb_2)}{2}, d = \frac{(tb_1 - ta_1) + (ta_2 - tb_2)}{2}$$

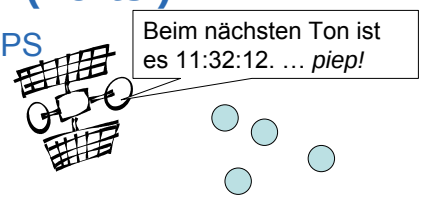
Uhren-Synchronisation (Forts.)

Hierarchisch



- Kindknoten synchronisieren mit Elternknoten (paarweise).
- O(n) Meldungen für Synchronisation
  - $O(n^2)$ wenn paarweise jeder mit jedem
- Hierarchie Konzept ähnlich zu Strata (NTP Protokoll aus dem Internet)

GPS



- Sensorknoten haben GPS Empfänger.
- Zeitinformation aus GPS Signal
- GPS Empfänger immer noch gross und teuer
- Mischformen GPS + hierarchische Synchronisation (wie NTP)

Problem #2: Speicher

- Speicher für Programm und Daten ist stark begrenzt.
- **Beispiel** Tmote Sky Node
  - 10 kB RAM für Program-daten (Stack, Heap)
  - 48 kB Flash-Speicher für Programmcode
  - 1 MBit serieller Flash-Speicher (z.B. für Messungen, nicht als Programmspeicher verwendbar)
- Nicht alles aus dem Desktop/Serverbereich ist auf Sensornetze übertragbar
- **Beispiel**
  - TCP/IP → nanoIP/nanoTCP (nanoIP+nanoTCP: 300B RAM, 1kB Flash)
  - Multithreading nur eingeschränkt möglich (Speicher für Thread-Context)

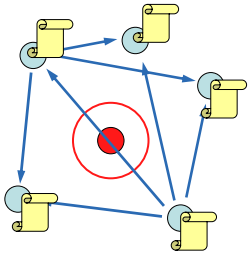
Problem #3: Drahtlose Übertragung

- Kommunikation
  - Drahtlos
  - (Enerie-)sparsam
  - Billige Komponenten
- Technologien
  - WLAN (802.11): meist zu teuer (Kosten/Energie)
  - Bluetooth (802.15.1): möglich, Protokoll komplex
  - „ZigBee“ (802.15.4): einfaches Protokoll, geringe Datenraten (theoretisch bis 250 kbps)
  - Proprietäre Lösungen

Drahtlos

- Interferenz unter den Knoten
- Störemissionen von aussen
  - Gleiches 2.4-GHz-Band für WLAN, Bluetooth, ZigBee, Garagentoröffner und Mikrowellengeräte(!)
- Fehlerrate bei Funkübertragung ist um ein Vielfaches grösser als z.B. bei Ethernet.
  - Carrier Sense (CA): möglich und sinnvoll
  - Collision Detection (CD): gleichzeitiges Senden und Empfangen schwierig

Übertragungs-Charakteristik (Experiment)



Setup
- 33 Knoten Testbed in CAB F31
- 1 Beacon-Knoten
- Gegenseitiges Senden von „Ping“ Nachrichten
- Pro Runde werden 32x31 = 992 Nachrichten versendet

Beacon-Knoten:
- Zu Beginn jeder Runde, Broadcast des Beacons mit max. Leistung an alle.

Nach Empfang des Beacons:
1. Stellen Runden-Uhr
2. Erzeugen Zufallspermutation der Liste aller (32-1) Knoten.
3. Senden „Ping“-Nachricht sequentiell an alle Knoten in der Liste.

Knoten *j* empfängt Ping von *i*
- Notieren sich Sender i in Sender-Vektor $\boldsymbol{s}_j$ (s_{ij} = 1)

Nach Ablauf der Runden-Uhr
- Loggen Sender-Vektor

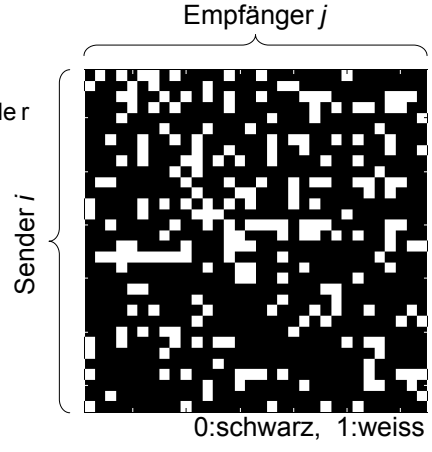
Adjazenz-Matrix

- Sender-Vektoren $\boldsymbol{s}_j$ bilden Adjazenz-Matrix $\boldsymbol{M} = (\boldsymbol{s}_1,\ldots,\boldsymbol{s}_{32})$
- $M_{ij} = s_{ij}$

$$M_{ij}^{(r)} = \begin{cases} 1 & \text{erfolgreiche Nachricht } i \rightarrow j \text{ in Runde } r \\ 0 & \text{sonst} \end{cases}$$

- Darstellung des Verbindungs-graphen: Kante von i→j wenn $\boldsymbol{M}_{ij}$=1.
- Diagonale jeweils 0 → Knoten senden keine Nachricht an sich selbst.

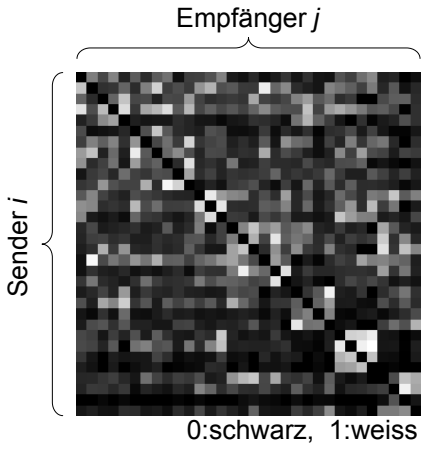
Graphische Darstellung von $M^{(r)}$



Adjazenz-Matrix (Forts.)

Arithmetisches Mittel

- Aus 170 Runden
- Empfangene Nachrichten (Mittel): 197.6/Runde
- Versendet: 32x31= 992/Runde
- Mittlere Übertragungs-Wsk: ≈0.2
- Matrix ist nicht symmetrisch!
  $|| \boldsymbol{M} ||_2 \approx 6$, $|| \boldsymbol{M} - \boldsymbol{M}^T ||_2 \approx 2$
  $i \rightarrow j \not\Rightarrow j \rightarrow i$



Beachte

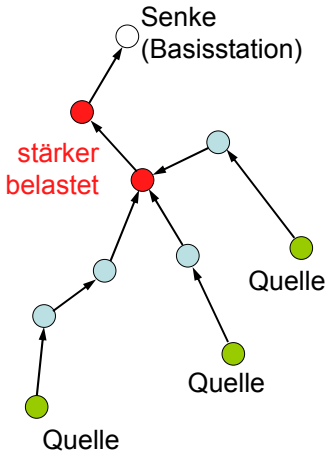
- Unrealistisches Kommunikations-Szenario: $O(n^2)$ Nachrichten
- Kein Scheduling der Nachrichten, nur Zufallspermutation
- Soll Problematik illustrieren

#4 Robustheit

- Sensor-Knoten sind unzuverlässig
  - Restkapazität der Batterien zu gering → **Fail-Stop**
  - Nachricht „verloren“ → **Omission**
  - Hardwaredefekt (Wassereinschluss, Feuchtigkeit, Vögel) → **Fail-Stop/Crash/ byzantinische Fehler**

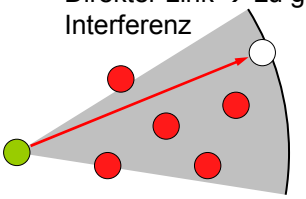
Beispiel

- Statischer (fest-codierter) Baum.
- 3 Quellen-Knoten erzeugen Datenstrom
- Knoten nahe der Senke stärker belastet als Knoten nahe einer Quelle
  → Brauchen grössere Batterie oder fallen schneller aus
- Statisch → kein Recovery → Fail-Stop
- Besser keine stat. Strukturen (ad-hoc)



Routing-Topologien in Sensornetzen

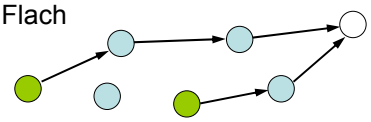
- Art der Kommunikation
  - Basisstation → Knoten
  - **Knoten → Basisstation**
  - Knoten → Knoten
- Ziel i.d.R. nicht direkt erreichbar
  - Zu weit entfernt/Sende-leistung zu gering
  - Direkter Link → zu grosse Interferenz



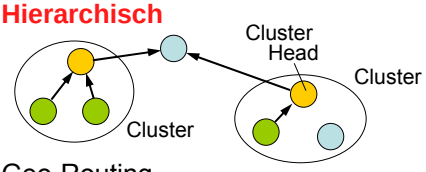
- Weiterleiten von Meldungen (**Multihop Routing**)

Netzwerk-Struktur

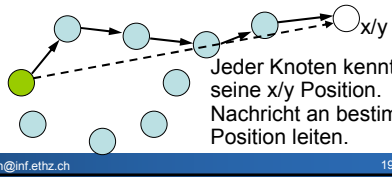
- Flach



- **Hierarchisch**

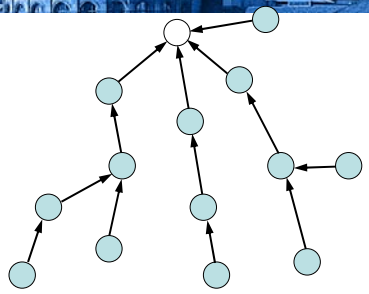


- Geo-Routing



Collection Tree Routing

- „Einsammeln“ von Daten an einer bestimmten Stelle.
- Sensor-Daten werden entlang eines Spannbaums zur Basisstation geschickt.
- Spannbaum:
  - **Wurzel**: Basisstation
  - **Blätter**: Datenquellen
  - **Innere Knoten**: Datenquellen und leiten Nachrichten weiter
- „Abfangen“ von Nachrichten auf dem Pfad zur Wurzel → Bündlung von Datenwerten, **Aggregation** (**data fusion**)



- Woher kommen die Pfeile?
  - Keine eigentlichen Kanten
  - Kantenreduzierter Verbin-dungsgraph? Realität?
- Dezentraler Algorithmus
  - Was sind die Inputs?
  - Stabiler Baum trotz Funk
  - Dynamisch (ad-hoc), Baum muss Störungen (Knoten, Interferenz) angepasst werden.

Verteilter Algorithmus für Spannbaum

Kurzfassung

Komponenten
- Messung der Link-Qualität zu „Nachbarknoten“
- Austausch und Management der Nachbar-Tabelle
- Weiterleiten von Nachrichten

Routing-Metrik (Gradient)
- Entfernung zum Wurzelknoten (Anzahl Hops)
- Oder: Erwartete #Übertragungen (ETX) bis zur Wurzel
  - Berücksichtigt Retransmissions infolge schlechter Links

Messung Links-Qualität
- Bidirektionale ETX-Werte
- ETX ~ #erwartete / #empfangene Nachrichten

Nachbar-Tabelle
`(NachbarID, Nachbars ETX zur Wurzel)`
- Periodischer Austausch der Tabelle
- Bei Empfang lokale Einträge entsprechend anpassen → Ziel: min ETX

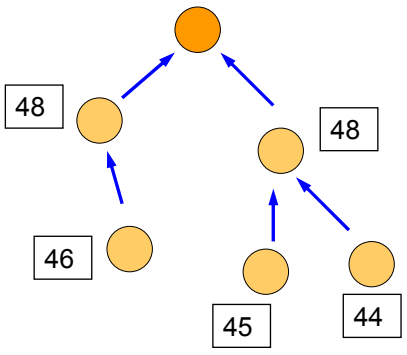
Weiterleiten von Nachrichten
- Senden an Nachbarn mit kleinstem ETX zur Wurzel

Daten-Aggregation

Was ist die *mittlere Temperatur* im Sensornetz?

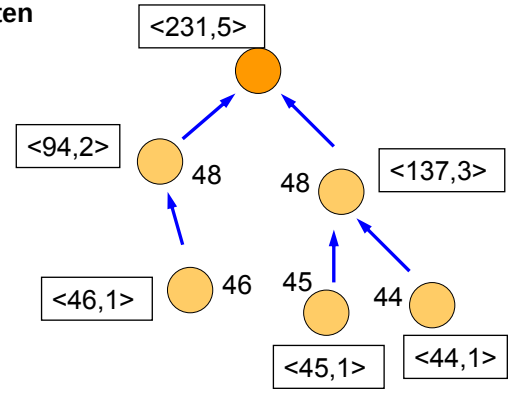
Aggregation in Basisstation

Mittelwert in Basisstation berechnet
Total # 8 Nachrichten

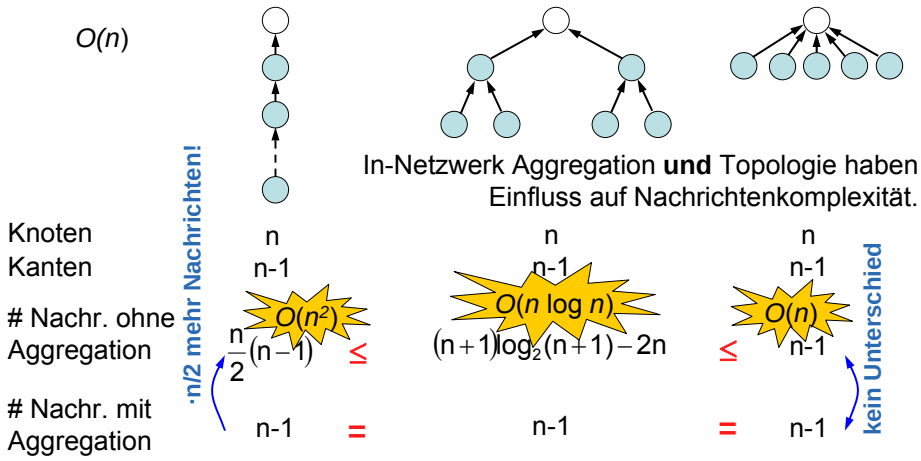


Aggregation im Netzwerk

Mittelwert berechnet aus 231/5 = 46.2
Total # 5 Nachrichten



Aggregations-Bäume



In-Netzwerk Aggregation

- Aggregation von räumlich verteilten, gleichzeitigen Messungen → **räumliche Aggregation (data fusion)**
- Mögliche räuml. Aggregate
  - Minimum (**`MIN`**)
  - Maximum (**`MAX`**)
  - Summe (**`SUM`**)
  - Mittelwert (**`AVG`**)
  - Varianz (**`VARIANCE`**)
  - Anzahl (**`COUNT`**)
  - Median
  - Histogramm

  } konst. Grösse
- In-Netzwerk Aggregation: → Idealerweise nur eine Nachricht/Kante

Aggregationszustand (AZ)

- Beinhaltet „Zusammenfassung“ der Werte für das Aggregat.
- Bsp: <Summe, Anzahl> für **`AVG`**
- Knoten senden Aggregationszustand in Richtung Wurzel
  - Vollständiger AZ: in Wurzel
  - Partieller AZ: Unterhalb Wurzel

Klassifikation

- Exemplarische Werte (**`MAX`**, **`MIN`**)
- Algebraisch (**`SUM`**, **`AVG`**, **`VARIANCE`**)
- Holistisch: Grösse ~ # Werte (Median, Histogram)

[Madden et al., OSDI'02]

Tiny Aggregation (TAG)

Distributive Aggregate f(<z>)
$f(<z_1> \cup <z_2>) = g(f(<z_1>), f(<z_2>))$
- Bsp: **MIN**, **SUM**, **AVG**

f(<z>) aus 3 Funktionen:
- **Initialisierer i**: <x> = i(x)
- **Merger m**: <z> = m(<x>, <y>)
- **Evaluator e**: z = e(<z>)

Bsp AVG
Zustand: <Summe, Anzahl>

$i: x \mapsto <x,1>$

$m: <s_1,c_1>,<s_2,c_2> \mapsto <s_1+s_2, c_1+c_2>$

$e: <s,c> \mapsto s/c$

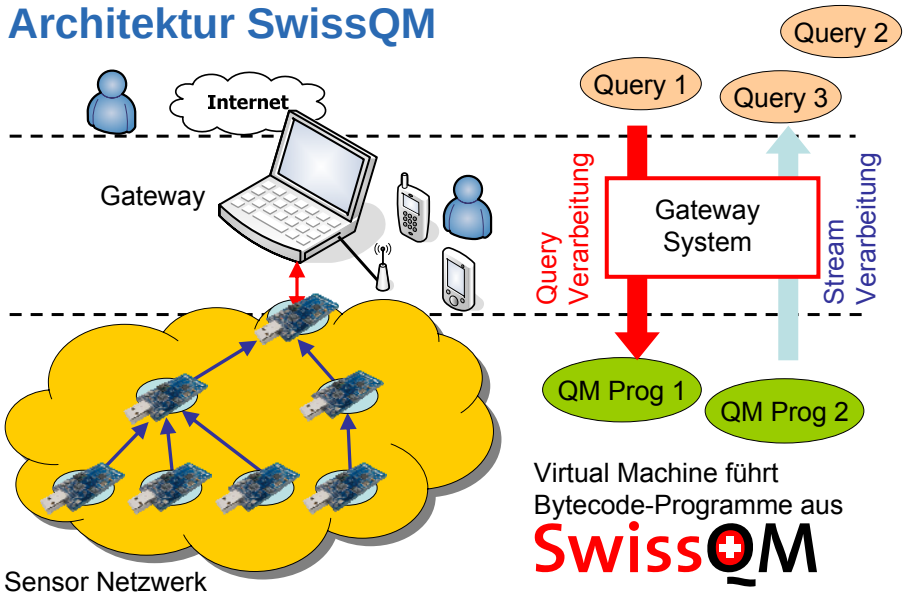
Implementation in Sensornetz
- Jeder Knoten *i*
  - Liest Sensor → x_i
  - Berechnet zughörigen part. Aggregationszustand $<z_i> = i(x_i)$
  - Empfängt $<z_j>,\ldots<z_k>$ von seinen direkten Kinder j,…,k
  - Berechnet

  $<z> = m(<z_i>, m(<z_j>, \ldots,<z_k>)))$
  - Sendet <z> an Elternknoten
- Basisstation
  - Empfängt vollständigen Zustand <z>
  - Wertet Aggregat aus: e(<z>)

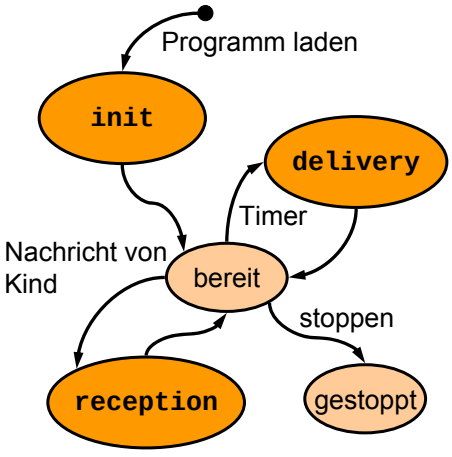
SwissQM – Demo

SwissQM

Architektur SwissQM



In-Netzwerk Aggregation mit SwissQM



- Datenstrom aus aggregierten Messungen
  - Code muss periodisch ausgeführt werden → Timer
- QM Programme bestehen aus 3 Code Abschnitten
  - Programm wird geladen (**init**)
  - Ein neues Tuple soll versendet werden (**delivery**)
  - Eine Nachricht wird von einem Kind empfangen/abgefangen (**reception**)

Vertiefende Vorlesungen und weitere Informationen

- 251-0380-00: Drahtlose Sensornetze, Kay Römer (FS08)
- 227-0557-00: Ad-hoc and Sensor Networks, Roger Wattenhofer (HS07)

- SwissQM Projekt
  http://swissqm.inf.ethz.ch

Werbung in eigener Sache… SwissQM

- Wir suchen immer motivierte Studenten für Projekte rund um SwissQM:
  - Bachelor-Arbeiten
  - Semesterarbeiten
  - Labor “Verteilte Systeme”
  - Master-Arbeiten
  - Hilfsassistenten

Kontakt

- René Müller, muellren@inf.ethz.ch

Projekt-Ideen

- **Multi-Tier Datenverarbeitung im heterogenes Netzwerk**
  - tmote sky + Linux Knoten
- **Power Management Issues im Linux Kernel**
  - Duty Cycling und Arbitrierung der Hardware Komponenten (Design)
- **Diskrete Eventverbeitung**
  - Programmieren von virtuellen Maschinen
  - Verteilte endl. Automaten
  - Events innerhalb und zwischen den Knoten
  - Actions/Event-Bedingungen in Bytecode