

Aggregierungsanfragen in Sensornetzwerken

Wenwo Huang

Betreuer: Heiko Schepperle



1. Sensornetzwerk Anwendung

das Erkennen und Vermeiden unfallträchtiger Situationen

Beispiel: Dichter Nebel ist eine mögliche Ursache von Verkehrsunfall

▪ Unterschiedliche Sensoren:

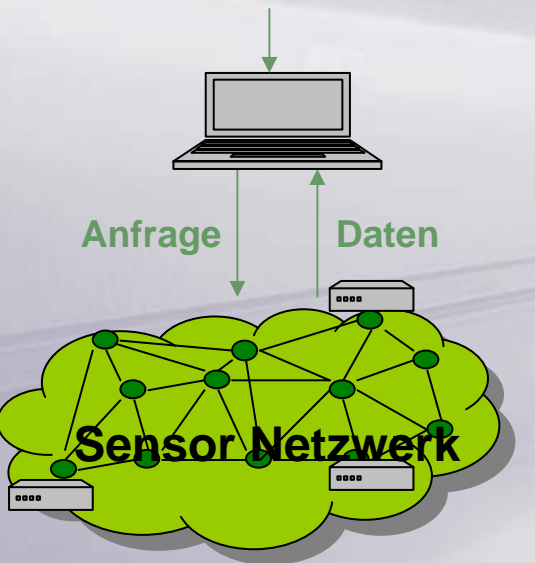
Licht (*Light*), Temperatur (*Temperature*), Luftdruck (*Air Pressure*),
Atmosphärische Feuchtigkeit (*Humidity*),
Geräusch (*Sound*), etc.

▪ Einzelne Daten von Sensoren sind uninteressant

→ Aggregation ist nötig.

▪ Ist der durchschnittliche Wert der allen Messwerten der Feuchtigkeit-Sensoren in einer Region A über einem Vordefinierten Grenzwert G?

```
SELECT AVG (S.value)
FROM Humidity Sensor S
WHERE S.loc IN Region A
Having AVG (S.value) >
Grenzwert G
```



Gliederung



1. Sensornetzwerk Anwendung

2. Grundlage

3. Einsatz 1: Tiny DB

4. Einsatz 2: Cougar

5. Fazit



2. Grundlage

2.1 Sensornetzwerke

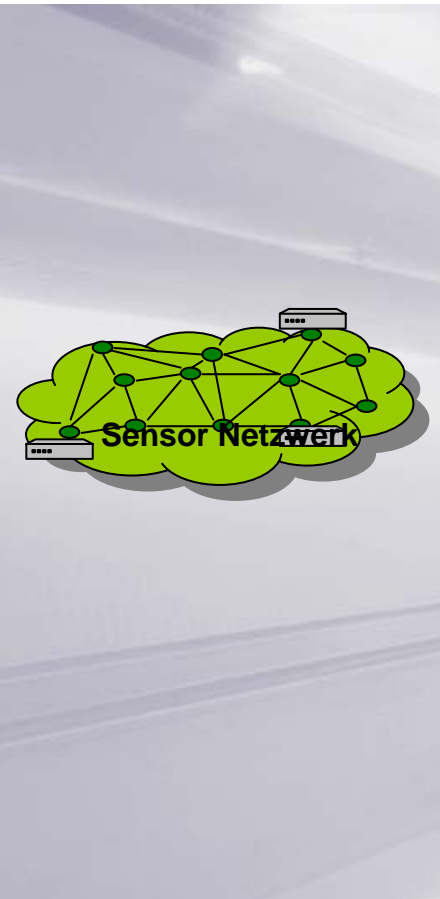
2.2 Anfragetypen

2.3 Allgemeine Aggregationstechniken



2.1. Grundlage - Sensornetzwerke

Definition



- Ein Sensornetzwerk besteht aus einer großer Anzahl von Sensorknoten.
- Einzelne Sensorknoten werden mit anderen Knoten in ihre Umgebung durch ein drahtlose Netzwerk verbunden.
- Sensorknoten kommunizieren sich mit den Knoten, welche räumlich entfernt sind, mit einem *multihop routing* Protokoll.
- Moderne Sensoren verfügen über einen eigenen Prozessor. Sie besitzen Berechnungsfähigkeit.



2.1. Grundlage - Sensornetzwerke

Einschränkung der Sensorressourcen



- Kommunikation
 - begrenzte Bandbreite und QoS
- Energieverbrauch
 - begrenzte Energieversorgung
- Berechnung
 - begrenzte Rechenleistung und Speicher
- Unsicherheit von Sensor-Messwerte
 - Unsicherheit wegen Einschränkungen in Sensor und Umweltgeräusch



2. Grundlage

2.1 Sensornetzwerke

2.2 Anfragetypen

2.3 Allgemeine Aggregationstechniken



2.2. Grundlage - Anfragetypen



- Historische Anfragen
 - Anfragen über die *historische Daten*, welche aus dem Sensornetzwerk bezogen wurden.
- Schnappschussanfragen
 - Die Schnappschussanfragen beschäftigen sich mit dem Netzwerk *in einem festgelegten Zeitpunkt*.
- Langlaufende Anfragen
 - Diese Anfragen beschäftigen sich mit dem Netzwerk *in einer Zeitstrecke*.

2. Grundlage

2.1 Sensornetzwerke

2.2 Anfragetypen

2.3 Allgemeine Aggregationstechniken



2.3. Grundlage - Allgemeine Aggregationsmethoden

Aggregation in Datenbank-Systeme

Aggregation in SQL-basiertem Datenbank-Systeme wird von einer *aggregate*-Funktion und einem *grouping*-Prädikat definiert.

- die standardmäßige *aggregate*-Funktionen:

COUNT, MIN, MAX, AVERAGE, SUM und zusätzlich Benutzer-definierte Funktionen (*UDFs: user-defined functions*).

- ein *grouping*-Prädikat teilt die Messwerte der Sensoren in Gruppe basiert auf einige Attribute auf.

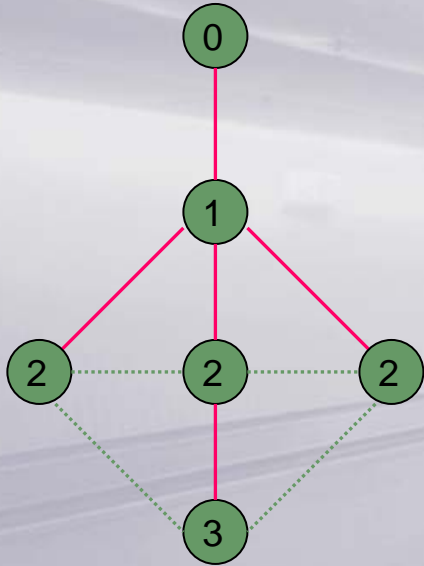
```
Beispiel: SELECT      TRUNC (temp/10), AVERAGE (light)
           FROM        sensors
           GROUP BY    TRUNC (temp/10)
           HAVING      AVERAGE (light) > 50
```



2.3. Grundlage - Allgemeine Aggregationsmethoden

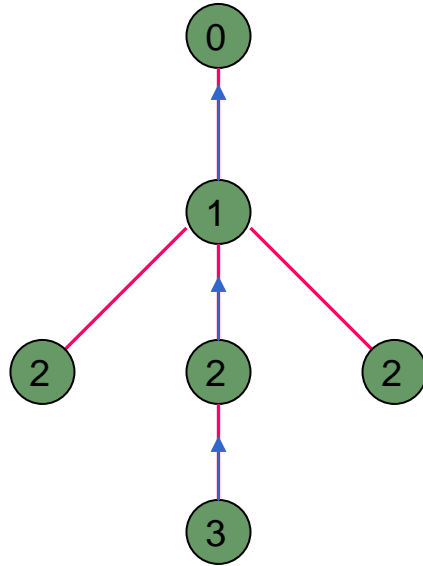
Beispiel a: Server-basierte Methode

Nachrichten:
 $3 + 6 + 1 = 10$

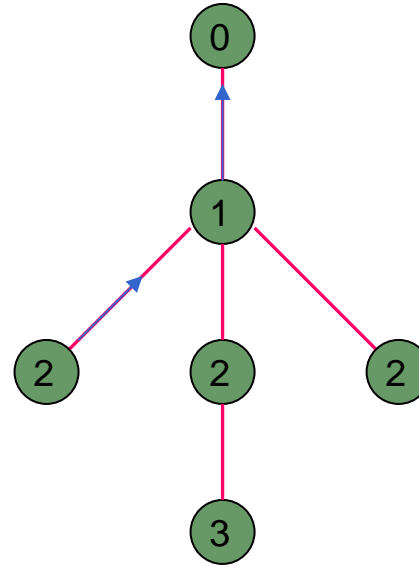


Routing-Baum

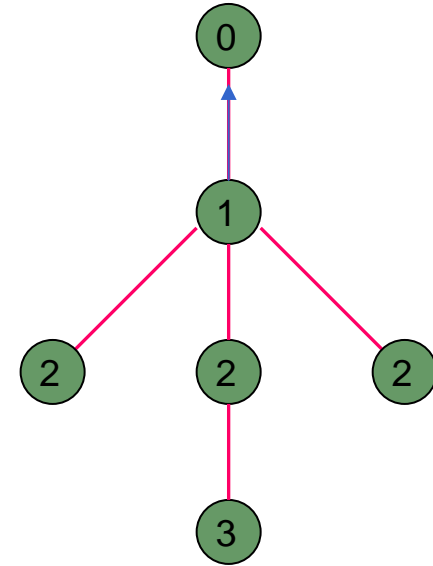
Alle Sensoren schicken ihre Daten entlang Multi-hop-Routen direkt zum Wurzel-Knoten (*root*) und berechnen sie an dem Wurzel-Knoten.



Nachrichten: Knoten 3: 3



Knoten 2: $2 + 2 + 2 = 6$

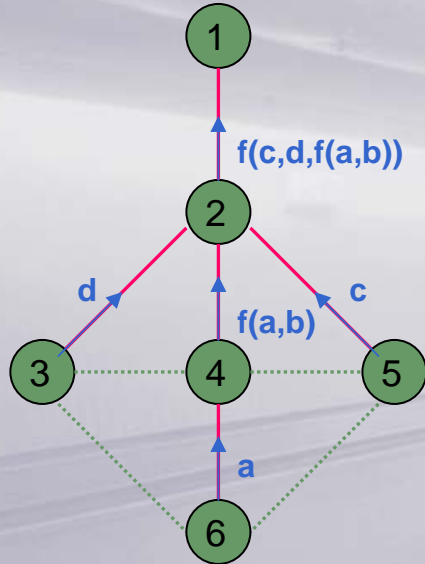


Knoten 1: 1

2.3. Grundlage - Allgemeine Aggregationsmethoden

Beispiel b: In-Netzwerk Methode

Nachrichten: 5



Routing-Baum

Grundidee:

Weil Kommunikation mehr Energie als Berechnung braucht, sollten die Verkehrsflüsse zwischen Knoten durch lokale Berechnung reduziert werden.

In-Netzwerk Methode:

Aggregate werden partiell oder völlig von die Sensoren selbst berechnet, während die Daten durch das Netzwerk zur host-PC leitet.

Nachrichten:

Insgesamt: $1 (a) + 1 (d) + 1 (f(a,b)) + 1(c) + 1 (f(c,d,f(a,b))) = 5$

Gliederung



1. Sensornetzwerk Anwendung

2. Grundlage

3. Einsatz 1: Tiny DB

4. Einsatz 2: Cougar

5. Fazit



3. Einsatz 1: TinyDB (Berkely Universität)

3.1 Überblick

3.2 Bearbeitung von Aggregierungsanfrage

3.3 Anfrageoptimierung



3.1. TinyDB - Überblick

Plattform: Motes + TinyOS

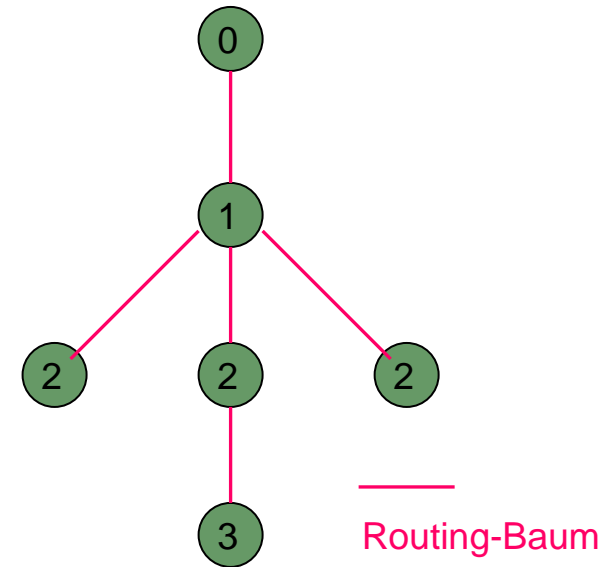
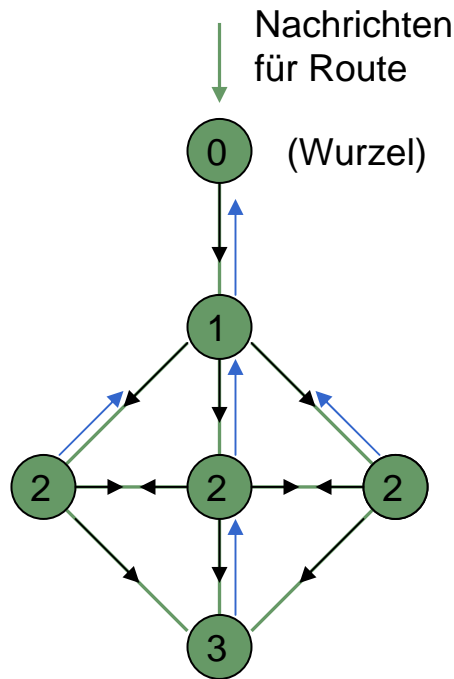
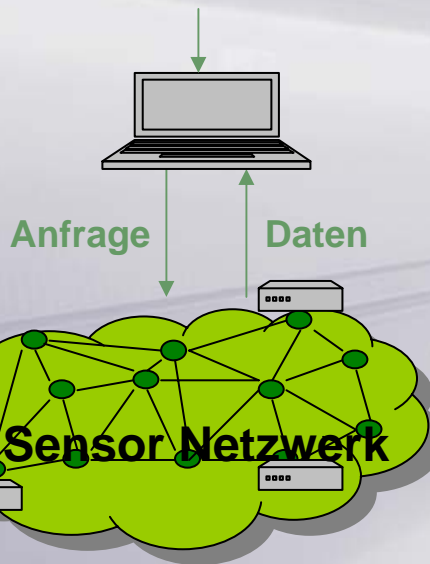
- Motes: kleine Sensorgeräte, besteht aus:
 - Speichern: 128KB Flash, 4KB RAM, 4KB ROM, 512 KB Externflash
 - einem Mikroprozessor: Amtel 8-bit 4 MHz
 - einem Funk (RF 916Mhz) und einer Menge von Sensoren
- TinyOS bietet die Service, welche die Schreibenprogramme vereinfacht. Diese Programme fangen die Sensordaten ab, bearbeiten sie und übertragen die Nachrichten über den Funk.
- Nachrichten in der aktuelle TinyOS-Generation:
 - 30 byte Nachrichten, Nachrichten id, Sensor id



3.2 TinyDB – Bearbeitung von Aggregierungsanfrage

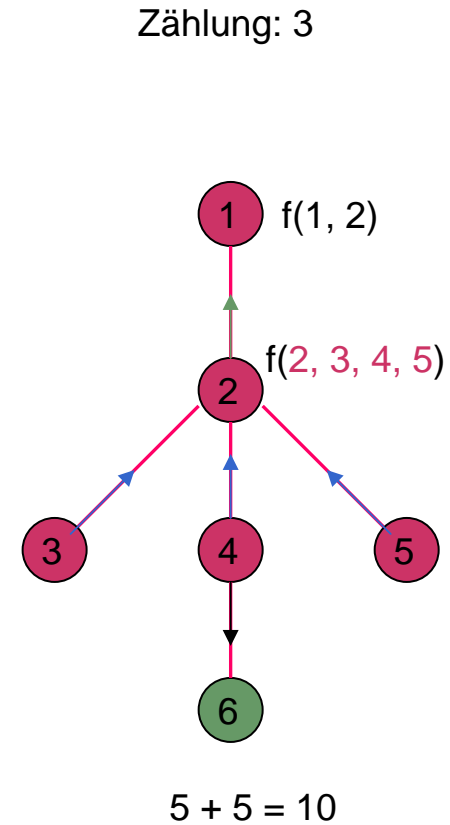
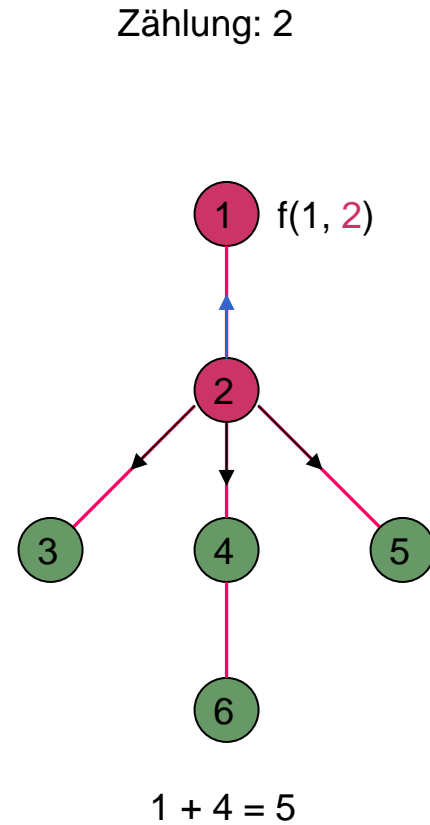
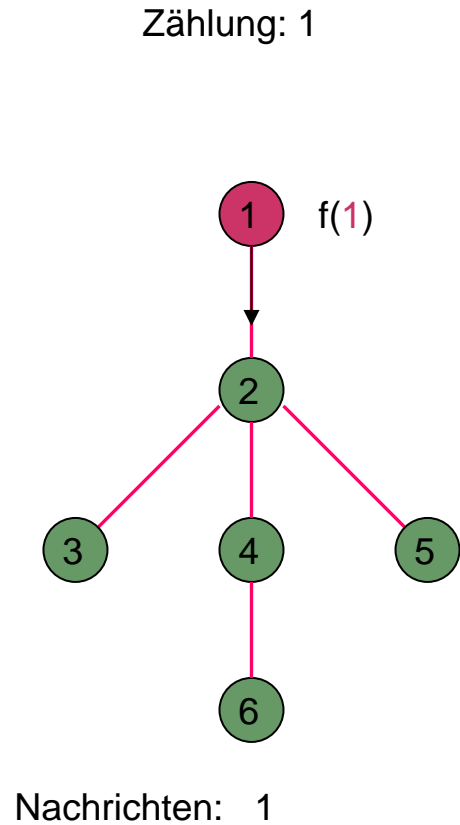
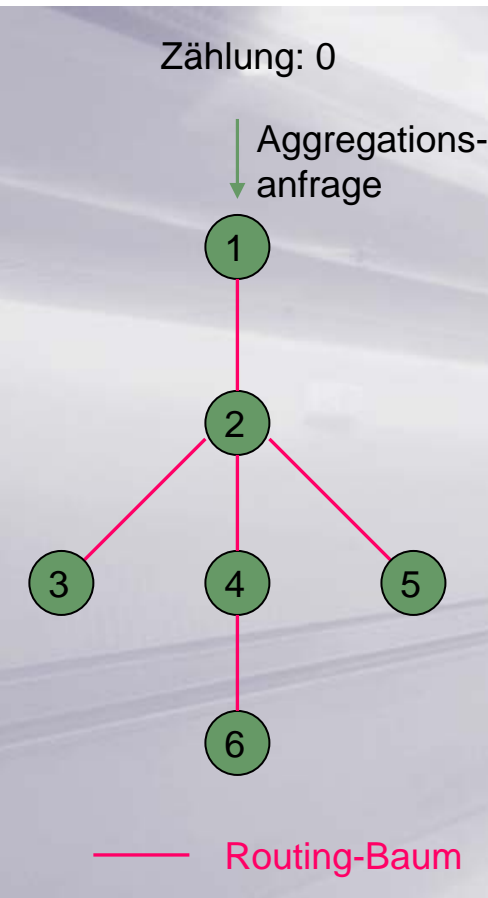
Routing Baum erzeugen

```
SELECT AVG (S.value)
FROM Humidity Sensor S
WHERE S.loc IN Region A
Having AVG (S.value) >
Grenzwert G
```



3.2 TinyDB – Bearbeitung von Aggregierungsanfrage

eine *Pipeline* Aggregation (1)



3.2 TinyDB – Bearbeitung von Aggregierungsanfrage

eine Pipeline Aggregation (2)

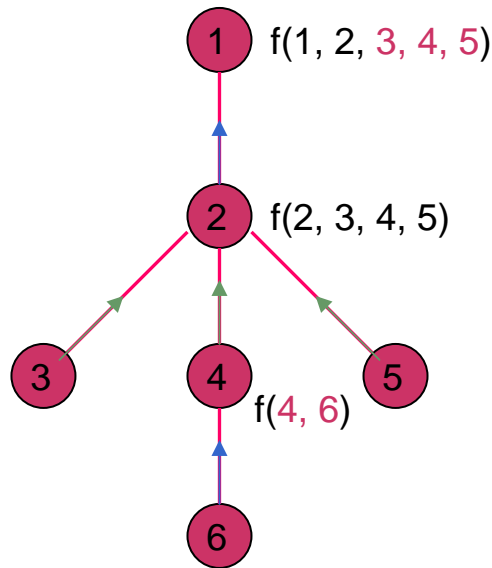
Nachrichten:
insgesamt 25



Anfrageoptimierung

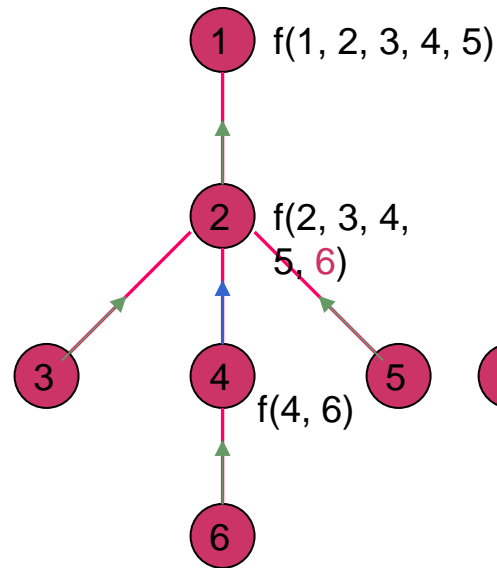
- Vermeiden der Duplikate
- Hypothese Testen

Zählung: 4



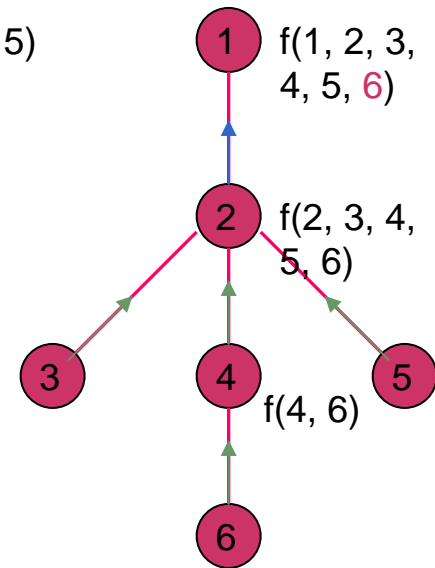
Nachrichten: $10 + 5 = 15$

Zählung: 5



$15 + 5 = 20$

Zählung: 6



$20 + 5 = 25$

3.3. TinyDB – Anfrageoptimierung

Vermeiden der Duplikate

Anfrageoptimierung

- Vermeiden der
Duplikate

- Hypothese

Testen

Grundgedanke:

- Sensoren übermitteln eine Nachricht, nur wenn der Aggregateswert verändert ist.
- Die Vaterknoten könnten annehmen, dass die Aggregateswerte ihrer Kinder unverändert sind, wenn sie keine neue Nachrichten von ihren Kinderknoten bekommen.

Für das vorne gegebene Beispiel, werden die unveränderte Nachrichten, die mit grünen Pfeilen gezeichnet wurden, nicht an Vaterknoten abgeschickt.

Nachrichten: insgesamt $25 - 12 = 13$



3.3 TinyDB – Anfrageoptimierung

Hypothese Testen

Anfrageoptimierung

- Vermeiden der Duplikate
- Hypothese Testen

Grundgedanke:

- Es wird nur die Sensordaten abgefangen, die das Resultat des Aggregate beeinflussen.
- Ein Sensor kann die Messwerte von in gleichen Ebene befindenden Sensoren aufspüren und schickt seinen eigenen Messwert nicht an Vaterknoten, wenn er weiß, dass sein Messwert das Resultat des Aggregate nicht beeinflussen kann.

Anwendbarkeit:

- ✓ MAX, MIN
- ~~SUM, COUNT~~



Gliederung



1. Sensornetzwerk Anwendung

2. Grundlage

3. Einsatz 1: Tiny DB

4. Einsatz 2: Cougar

5. Fazit



4. Einsatz 2: Cougar (Cornell Universität)

4.1. Überblick

4.2. Anfrageplan

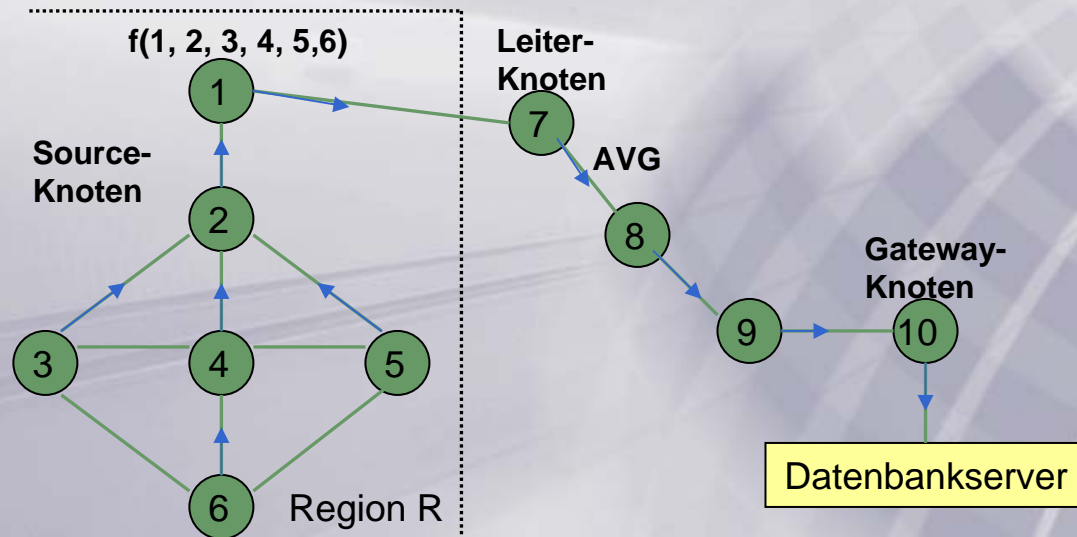
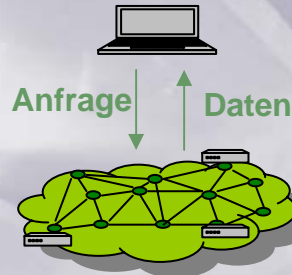
4.3. Bearbeitung von Aggregierungsanfrage



4.1. Cougar - Überblick

Sensornetzwerk Model

```
SELECT AVG (S.value)
FROM Humidity Sensor S
WHERE S.loc IN Region A
Having AVG (S.value) >
Grenzwert G
```

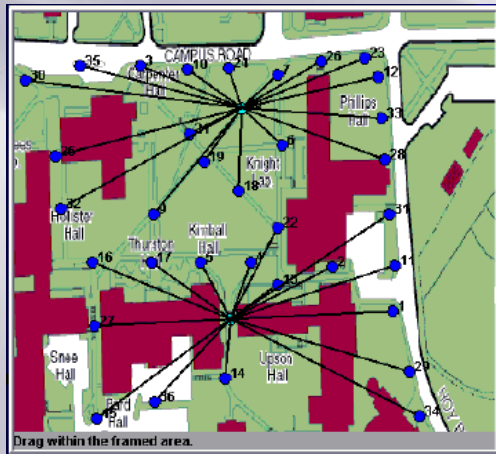


Sensoren:

- **Source-Knoten:** liefern nur Daten
- **Leiter-Knoten:** die Daten von den ihnen unterstellten Sensoren zu sammeln und mit diesen Daten berechnen. Die Ausgangsdaten der Sensoren müssen nur zum Leiter-Knoten geschickt werden und nicht zum Gateway-Knoten
- **Gateway-Knoten:** Alle Kommunikationen von außerhalb des Sensornetzwerks müssen durch den Gateway-Knoten durchgehen.

Sensor-Daten: Datensatz mit einigen Feldern: id, Standort, Zeitstempel, Sensortyp, abgelesener Messwert

4.2. Cougar – Anfrageplan (1)



- Ein Anfrageplan entscheidet, wie viele Berechnungen im Netzwerk durchgeführt werden, und bestimmt die Rolle und die Verantwortlichkeit jedes Sensorknotens, und spezifiziert, wie die Anfrage durchgeführt werden und wie die relevante Sensoren koordiniert werden.
- Ein Anfrageplan wird in sogenannte Flussblöcke (*flow blocks*) zerlegt, Ein Flussblock ist eine Menge von Sensoren mit einem Leiter-Knoten.



4.2. Cougar – Anfrageplan (2)

Um die Sensordaten zum Leiter zu schicken gibt es verschiedene Möglichkeiten:

- Direktes Verschicken der Daten an den Leiter
- Zusammenlegung von Paketen (Aus vielen kleinen wird ein großes Paket zusammengelegt)
- Partielle Berechnung durch Unterknoten (Die Daten werden auf dem Weg schon aufbereitet durch eine Art Unter-Leiter-Knoten)



4.3. Cougar - Bearbeitung von Aggregierungsanfrage

Synchronisation (1)

In-Netzwerk Aggregation:

- Zusammenlegung von Paketen
- Partielle Berechnung durch Unterknoten



Synchronisation

Die Aufgabe der Synchronisation zwischen den Knoten des Netzes sind:

- auf wie viele Sensormesswerte zu warten und wann *Zusammenlegung von Paketen* oder *Partielle Berechnung durch Unterknoten* ausgeführt wird
- das Warten auf nicht erreichbare Knoten zu vermeiden.



4.3. Cougar - Bearbeitung von Aggregierungsanfrage

Synchronisation (2)

Synchronisation:

- Incremental Time Slot
- Vorhersagen durch
Vergangenheitsbetrachtung

Incremental Time Slot

Jeder Knoten im Netz wartet eine bestimmte Zeit, bevor er entscheidet, ob die ihm untergeordneten Knoten erreichbar sind.

Probleme:

- die Wartezeit im Voraus zu bestimmen ist sehr schwierig.
- bei regelmäßigen Ausfällen und somit regelmäßigem Umstrukturieren des Baumes müssen allen Sensoren neue Wartezeiten zugewiesen werden.
- die Sensoren sind niemals ganz zeitsynchron, wenn keine aufwendige Zeitsynchronisationsprotokolle benutzt werden.

→ Lösung: Vorhersagen durch Vergangenheitsbetrachtung



4.3. Cougar - Bearbeitung von Aggregierungsanfrage

Synchronisation (3)

Synchronisation:

- Incremental Time Slot
- Vorhersagen durch
Vergangenheitsbetrachtung

Vorhersagen durch Vergangenheitsbetrachtung

Sobald ein Knoten p ein Paket von Knoten n erhält, wird n auf die Warteliste von p gesetzt. p (Vaterknoten von n) erwartet, andere Pakete von n in der nächsten Runde zu empfangen.

Die somit von p getroffene Vorhersage kann auf zwei Weisen fehlschlagen:

- Knoten n hat einen neuen Vaterknoten, dann ist das Löschen von n aus der Warteliste von p richtig.
- n sendet seine Daten nicht an p , weil eine lokale Bedingung nicht erfüllt ist z.B. ein Mindestgrenze. Das Löschen von n aus der Warteliste von p ist dann aber falsch.
—→ Lösung: muss n seinen Vaterknoten p benachrichtigen, dass es keine Daten senden wird.

Gliederung



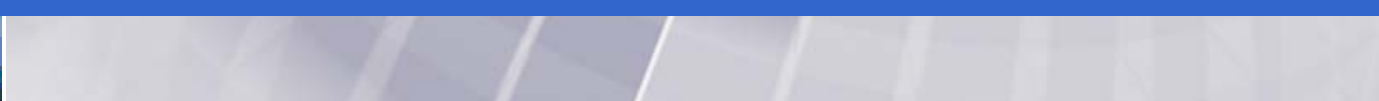
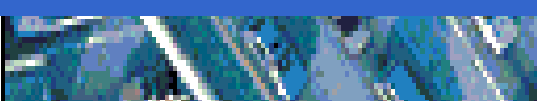
1. Sensornetzwerk Anwendung

2. Grundlage

3. Einsatz 1: Tiny DB

4. Einsatz 2: Cougar

5. Fazit



5. Fazit

- Sensoren haben die beschränkte Ressourcen.
- Einzelne Daten von Sensoren sind uninteressant. Aggregation ist nötig.
- Weil Kommunikation mehr Energie als Berechnung braucht, sollten die Verkehrsflüsse zwischen Knoten durch lokale Berechnung reduziert werden (In-Netzwerk Aggregation).
- **TinyDB:** In-Netzwerk Aggregationsmethode: eine *Pipeline* Aggregation
 Optimierung → 1) Vermeiden der Duplikate 2) Hypothese Testen
- **Cougar:** In-Netzwerk Aggregationsmethode: 1) Zusammenlegung von Paketen
 2) Partielle Berechnung durch Unterknoten → Synchronisation:
 Incremental Time Slot Optimierung →
 Vorhersagen durch Vergangenheitsbetrachtung



Literatur

- Samuel Madden, Robert Szewczyk, Michael j. Franklin und David Culler; Supporting Aggregate Queries Over Ad-Hoc Wireless Sensor Networks; Seite 49, ISBN:0-7695-1647-5, 2002
- Yong Yao und Johannes Gerke; Query Processing for Sensor Networks; <http://www.cs.cornell.edu/johannes/papers/2003/cidr2003-sensor.pdf>
- Raymond Pon und Wesley W. Chu; Challenges and Issues in Querying Sensor Networks;
- Wei Hong und Sam Madden; Implementation and Research Issues in Query Processing for Wireless Sensor Networks; <http://csdl.computer.org/comp/proceedings/icde/2004/2065/00/20650876.pdf>
- Björn Christmann; Verarbeitung von Sensordaten – Seminar Data Warehousing im Verkehrsbereich in SS 2003 von Institut IPD, Uni Karlsruhe;



Aggregierungsanfragen in Sensornetzwerken

Fragen?

