Update-Linearisierbarkeit

Ein Konsistenzkonzept für Sensornetze

Else

11. Dezember 2012

Outline

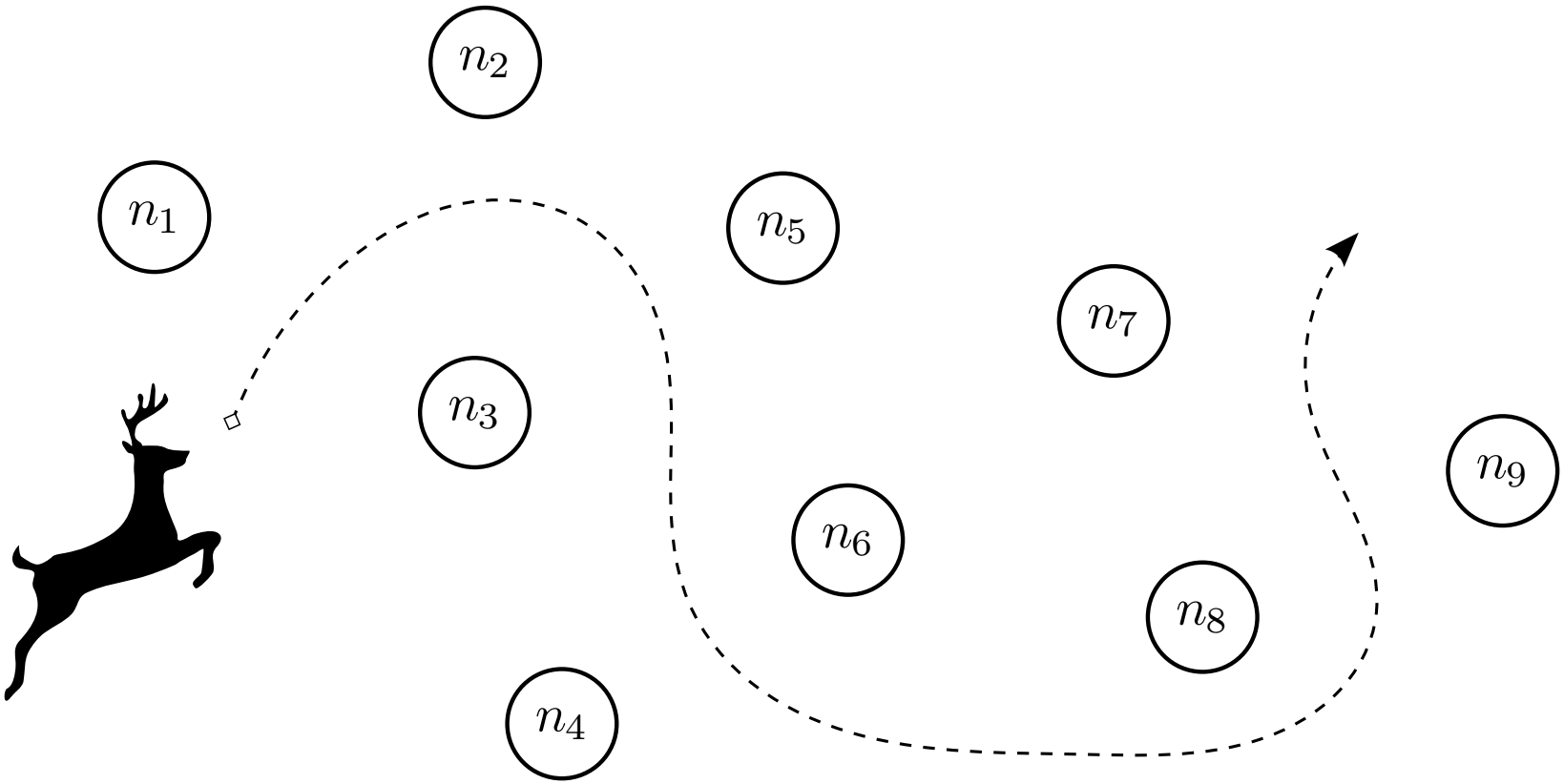
Sensor Nodes

Abbildung: BTnode (source: [ETHZ])

- 8bit Atmel Prozessor @8MHz
- 4kByte EEPROM, 64kByte SRAM, 128kByte Flash
- zwei Funkschnittstellen
  - Bluetooth (ISM, 2.4GHz)
  - low-power 868MHz
- max. Stromverbrauch:
  - 198 mW / 102.3 mW

Einsatzgebiete

Unser Beispiel: Bewegungen erkennen



Ereignisse sortieren

- Alternative 1: jedem Event Timestamp zuordnen
  - $t(e_1) < t(e_2) \Rightarrow e_1 \xrightarrow{\text{before}} e_2$
- aber: erfordert "globale Zeit" $\rightarrow$ clock synchronization
- Alternative 2: "logische" Uhren
  - e.g. Vector clocks
  - Vektor der Größe n mit logischer Uhr pro Prozess
- aber: globale Prozesssicht erforderlich; anfällig für Partitionierungen

Die “occured-before“ Relation

- definiert die Ordnungsgenauigkeit $\delta > 0$
  - minimaler zeitlicher Abstand zwischen zwei Events
- Beispiel: zwei Events u and u'
  - zeitlichen Abstand berechnen: $t_{obs}(u') - t_{obs}(u)$
    - wenn Abstand $> \delta$: u *occured-before* u'
    - wenn Abstand $< \delta$: u und u' sind *concurrent*
- benötigt keine synchronisierten Uhren (t_{obs} lokal)
- kein besondere Overhead für Observer (energieeffizient)

Terminologie

Objekte mobile oder stationäre, wahrnehmbare Objekte
Beispiel: Eintagsfliege mit RFID Halsband.

Terminologie

Objekte mobile oder stationäre, wahrnehmbare Objekte
Beispiel: Eintagsfliege mit RFID Halsband.

Observer Geräte, die Objekte wahrnehmen können
Beispiel: wireless sensor node

Terminologie

Objekte mobile oder stationäre, wahrnehmbare Objekte
Beispiel: Eintagsfliege mit RFID Halsband.

Observer Geräte, die Objekte wahrnehmen können
Beispiel: wireless sensor node

Update-Requests Statusnachrichten vom Observer, welche zum DB Node gesendet werden

Terminologie

Objekte mobile oder stationäre, wahrnehmbare Objekte
Beispiel: Eintagsfliege mit RFID Halsband.

Observer Geräte, die Objekte wahrnehmen können
Beispiel: wireless sensor node

Update-Requests Statusnachrichten vom Observer, welche zum DB Node gesendet werden

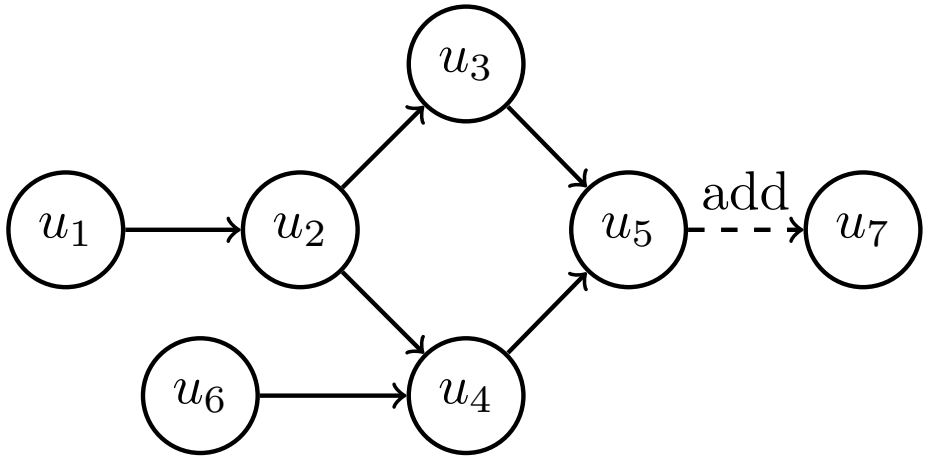
DB Node Empfängt, verarbeitet und speichert Update-Requests

Update-Linearisierbarkeit

- S_x: Serialisierung von Lese- und Update-Operationen für Objekt x
- S_x update-linearisierbar, wenn
  1.
     - Lese-Operationen in der Reihenfolge des Client-Programms sind
     - Alle Kombinationen von Update-Requests gemäß occured-before geordnet sind
  2.
     - S_x darf nach außen nicht seinen verteilten Charakter zeigen

Speicherung der Ordnungsinformationen

- Datenstruktur zum Speichern der occured-before Relation
- $G_x = (V_x, E_x)$
  - V_x: lokal bekannte Update-Requests
  - E_x: lokal bekannte Ordnungsrelation zwischen Update-Requests
- eine Kante existiert von a nach b, wenn: a occured-before b



Operationen: Neuen Knoten hinzufügen

- bei Ankunft eines neuen Update-Requests beim DB Node muss Ordnungsgraph aktualisiert werden
- Update-Requests wird akzeptiert, wenn alle Knoten occured-before sind
  - neuer Update-Request wird zur Menge der Knoten hinzugefügt
  - alle existierenden Knoten bekomme eine neue Kante zum neuen Knoten
- ansonsten wird der Update-Request zurückgewiesen

Operationen: Zwei Graphen joinen

- wenn der Ordnungsgraph geändert wurde, müssen diese Informationen weitergegeben werden
- DB Nodes verteilen Ordnungsinformationen durch Austausch von Graphen
- wenn ein DB Node einen solchem empfängt, wird der Graph mit dem existierenden zusammengeführt

Update-Entscheidungen ableiten

- wenn ein DB Node einen Update-Request für ein Objekt erhält, muss er entscheiden, ob die Datenbank aktualisiert wird oder nicht

- Entscheidungen werden getroffen durch Analyse des Ordnungsgraphen
- Entscheidung wird
  - akzeptiert, wenn Nachricht vom selben Knoten und die Sequenznummer größer ist
  - akzeptiert, wenn Nachricht vom Knoten des Datenbankeintrags erreichbar ist
  - ansonsten verworfen

Observer-Node Protokoll

- wenn Observer neuen Objektzustand bemerkt, erhöht er seine Sequenznummer und leitet O-Update Nachricht an DB Nodes weiter
  - O-Update: (Sequenznummer, Update-Request)
- bei Empfang: DB Node akzeptiert neuen Zustand, wenn der letzte Update-Request länger als δ Zeit her ist
- DB Node aktualisiert den Ordnungsgraphen und sendet eine N-Update Nachricht zu anderen DB nodes (Node-Node Protokoll)
  - N-Update: (Ordnungsgraph, Update-Request)

Node-Node Protokol

- DB Node muss Entscheiden, ob neuer Zustand akzeptiert wird
- Entscheidung gemäß folgendem Schema getroffen:
  - akzeptieren, wenn es noch keinen Objektzustand gibt
  - sonst: wenn es einen Objektzustand gibt, der
    - vom selben Observer ist: akzeptieren, wenn Sequenznummer größer ist
    - von einem anderen Observer ist: akzeptieren, wenn letztes Update occurred-before

Zusammenfassung

- ermöglicht chronologische Ordnung von verteilten Events
- eher geringer Overhead für Observer $\rightarrow$ energieeffizient
- geeignet für Szenarien mit mehr Lese- als Schreib-Operationen

- Zeitlicher Vergleich von mehreren Objekten nicht möglich
- nicht geeignet für Anwendungen mit geringen Anforderungen an Latenz
- benutzt Flooding (ineffizient)

Danke.

Quellen

BTnode http://www.btnode.ethz.ch/

Mattern, Friedemann
Die Informatisierung des Alltags - Leben in smarten Umgebungen, 2007
Springer Verlag, 1. Auflage, ISBN 9783540714545

Hähner, J., Rothermel, K., & Becker, C. (2004). Update-linearizability: a consistency concept for the chronological ordering of events in MANETs. 2004 IEEE International Conference on Mobile Adhoc and Sensor Systems IEEE Cat No04EX975 (pp. 1–10). IEEE. doi:10.1109/MAHSS.2004.1392060