Zeitsynchronisation in Sensornetzen

Seminarvortrag von

Tobias Krauer

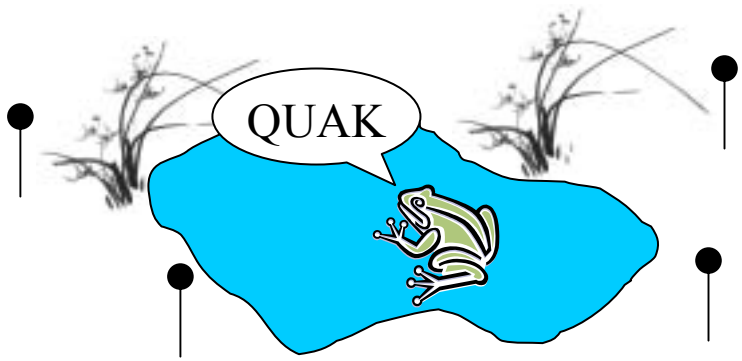
Gliederung

1. Motivation & Einführung
2. Anforderungskatalog
3. Verfahren in traditionellen verteilten Systemen
4. Packet Stream Synchronization
5. Reference Broadcasts Synchronization
6. Schlussbetrachtung

Motivation

Anwendungsbeispiel Zeitsynchronisation:

- „Data Fusion“, z.B. bei Froschbeobachtung

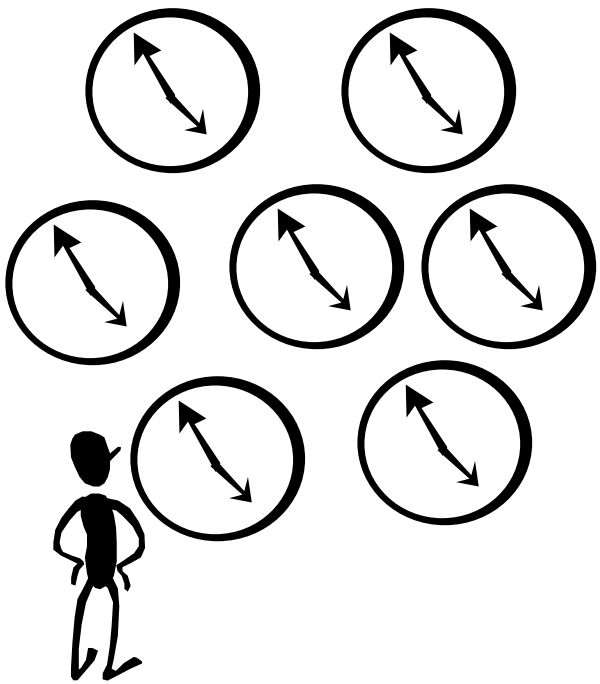


- Position des Frosches bestimmen
- Unterschiede der Signallaufzeit auswerten
- Dafür müssen Uhren synchronisiert sein
- Für cm-genaue Ortung ist bereits Präzision im µs-Bereich nötig

- Weitere Anwendungen: Energiesparende Kommunikation, Aufgaben wie in traditionellen verteilten Systemen: Security, Scheduling,...

Was ist Zeitsynchronisation?

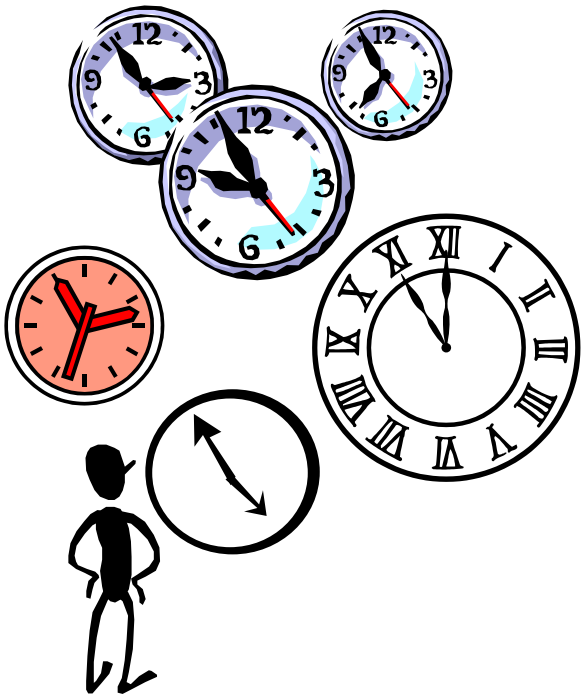
Wunsch-Zustand:



- Alle Uhren zeigen gleiche Zeit
- Laufen immer synchron
- Alle Uhren zeigen Realzeit (UTC) an
  - Nicht unbedingt notwendig
  - „Externe Synchronisation“

Was ist Zeitsynchronisation?

Ist-Zustand:



- Alle Uhren zeigen andere Zeit
- Alle Uhren haben unterschiedliche Frequenz
- Frequenz ist nicht stabil
  - ➢ abhängig von Temperatur, Spannung,...

Gliederung

1. Motivation & Einführung
2. Anforderungskatalog
3. Verfahren in traditionellen verteilten Systemen
4. Packet Stream Synchronization
5. Reference Broadcasts Synchronization
6. Schlussbetrachtung

Anforderungskatalog

Anforderungen seitens Sensornetz:

- Energiesparend
- Eigenheiten der Topologie Rechnung tragen:
  - Dynamisch, ad hoc, verbindungslos, Knoten zeitweise unerreichbar
- Hohe Präzision (µs-Bereich)
- Sensorknoten sollten klein und billig sein
- Skalierbar und robust

Anforderungskatalog

Anforderungen seitens Zeitsynchronisation:

- Uhr darf nie zurückgestellt werden
  - ➢ Sonst zweimal gleicher Zeitpunkt!
- Instabilität der Frequenz („clock drift“) ausgleichen
  - ➢ Uhrendriftrate ca. 10^{-6} (1s in 10 Tagen)

Gliederung

1. Motivation & Einführung
2. Anforderungskatalog
3. Verfahren in traditionellen verteilten Systemen
4. Packet Stream Synchronization
5. Reference Broadcasts Synchronization
6. Schlussbetrachtung

Traditionelle Methoden

In traditionellen verteilten Systemen sind bereits unterschiedliche Methoden im Einsatz:

- GPS
- Logische Uhren (Lamport, Vektorzeit)
- Network Time Protocol (NTP)

Network Time Protocol

- David L. Mills, anfangs 80er Jahre
- Grundidee: Versenden von Nachrichten mit Zeitstempel, Latenz wird mit halbierten round-trip-time geschätzt
- Hat sich seit Jahren im Internet bewährt
- Sehr robust, skalierbar (→ Hauptkriterium)
- Hierarchische Baumstruktur (Stratum Server)
- Genauigkeit: Im ms-Bereich

Network Time Protocol

Wieso nicht NTP in Sensornetzen verwenden?

- Zu ungenau
  - ➢ Kleine symmetrische Latenz als Voraussetzung
- Dauerndes horchen am Netz ist reine Energie-verschwendung
- Knoten sind nicht immer erreichbar
  - ➢ Erst später synchronisieren nicht möglich!
- Setzt statische baumartige Infrastruktur voraus

Gliederung

1. Motivation & Einführung
2. Anforderungskatalog
3. Verfahren in traditionellen verteilten Systemen
4. Packet Stream Synchronization
5. Reference Broadcasts Synchronization
6. Schlussbetrachtung

Packet Stream Synchronization

Clock Synchronization using Packet Streams

Philipp Blum, Lothar Thiele

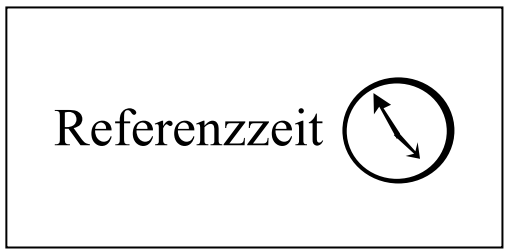
Institut für Technische Informatik und Kommunikationsnetze, ETHZ

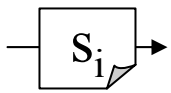
Grundideen:

- Ausgewählter Knoten, der einen Stream von Zeitstempeln generiert
- Verwendung von Software-Uhren
- Minimaler Delay relativ konstant

Systemmodell

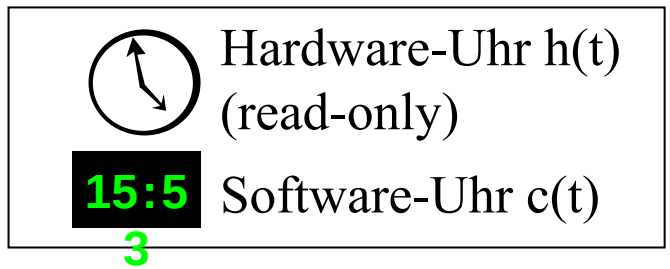
Sender = „Produzent“:





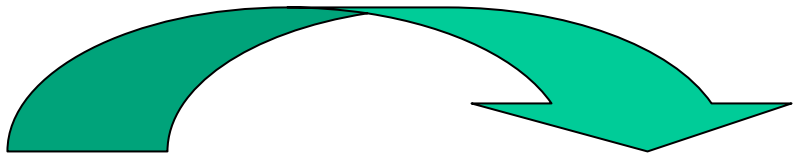
Zeitstempel i

Empfänger = „Konsument“:



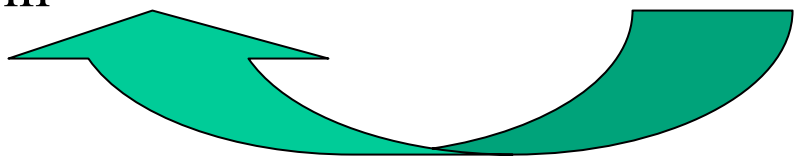
- Verzögerung ($s_i - r_i$) ist variabel, keine normierte Verteilung
- Beobachtung: Minimaler Delay ist relativ konstant
  - ➢ Unabhängig von Netzlast und Empfänger

Das Grundproblem



Um Zeit synchronisieren zu können, muss die Verzögerungszeit der Kommunikation bekannt sein

Um die Verzögerungs-zeit der Kommunikation zu messen, müssen die Uhren synchronisiert sein



- Wird bei diesem Algorithmus durch die Annahme Latenz = 0 gebrochen

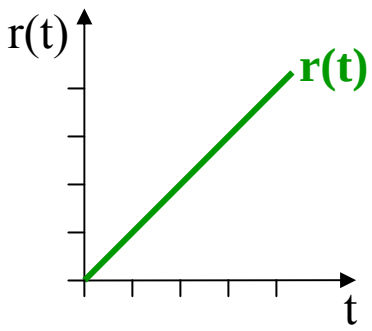
Uhrendrift

Ein weiteres Problem, der Uhrendrift:

- Verantwortlich, dass Synchronisation dauernd wiederholt werden muss
  - ➢ Hochpräziser Algorithmus müsste also dauernd Nachrichten senden!
- Definition des Uhrendrifts:

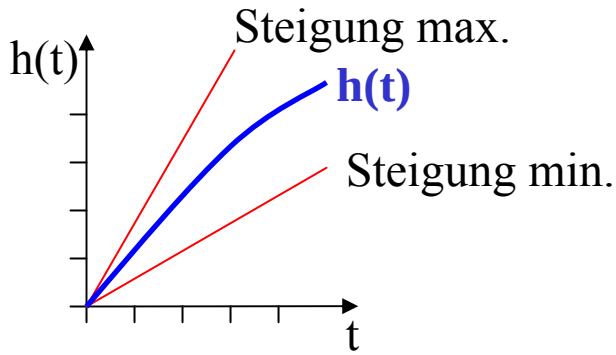
$$\rho^h(t) = \frac{\partial h(t)}{\partial t} - 1 \qquad \text{Forderung}: \left|\rho^h(t)\right| < \rho^h_{\max}$$

Uhrendrift



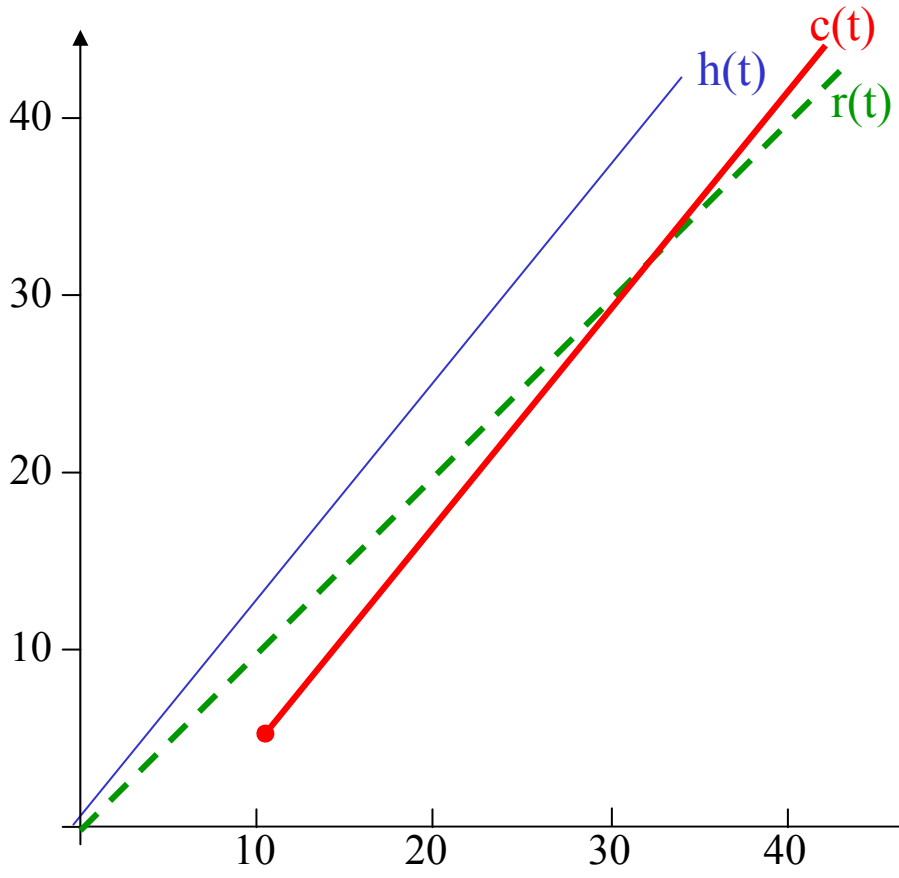
Referenzzeit r(t) ist eine Gerade mit Steigung 1

- Der Uhrendrift ist immer gleich 0



Die Funktion der Hardware-Uhr ist eine Funktion mit einer maximalen Steigung $(1+\rho^{h}_{max})$ und einer minimalen Steigung $(1-\rho^{h}_{max})$

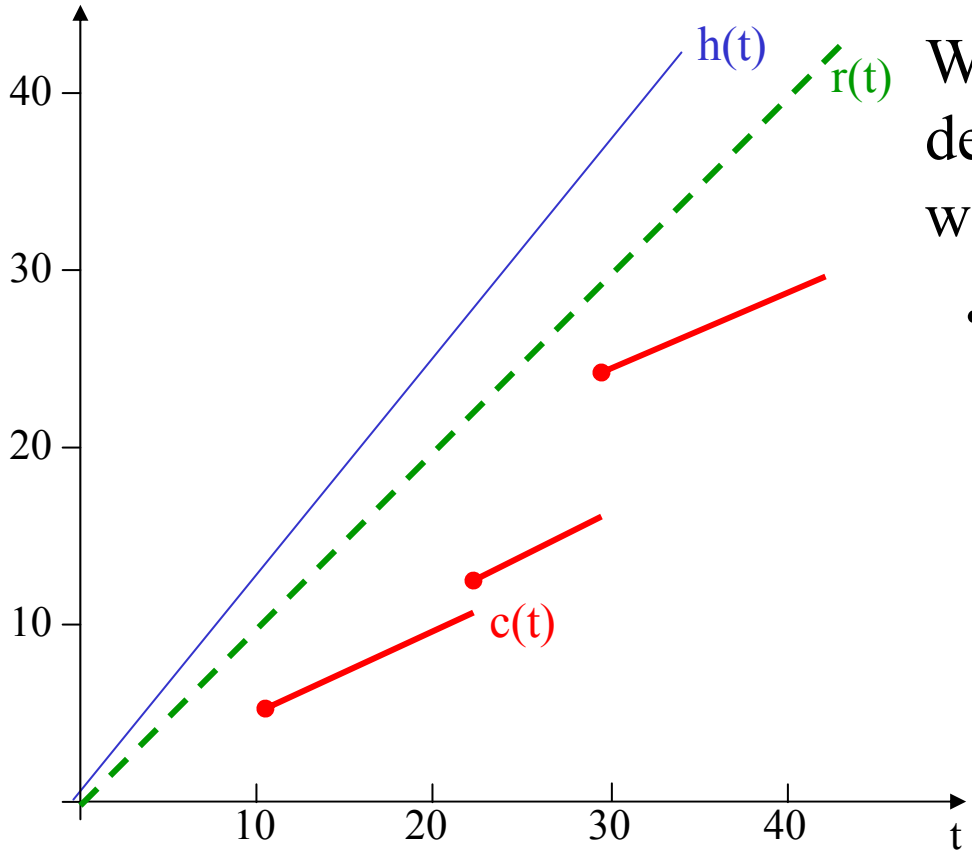
Wie weiter?



Was soll mit der Information des Zeitstempels gemacht werden?

- Software-Uhr parallel zur Hardware-Uhr laufen lassen?
  - ➢ Sind neue Zeitstempel nicht genauer?
  - ➢ Gefahr das Software-Uhr potentiell unendlich abdriftet!

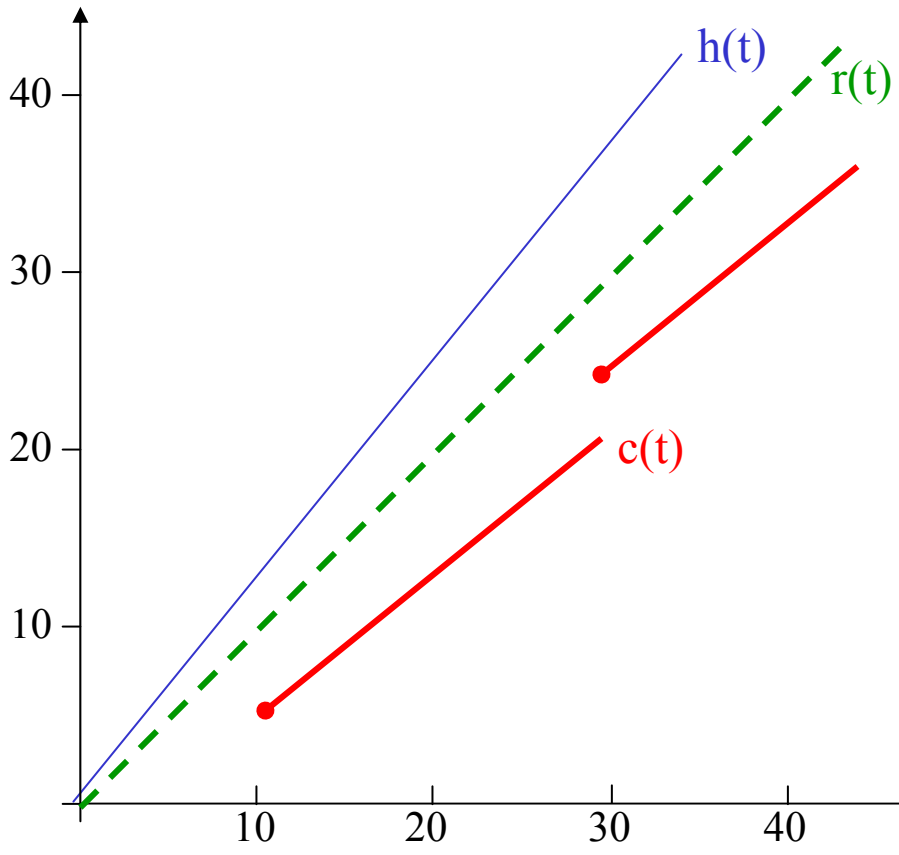
Wie weiter?



Was soll mit der Information des Zeitstempels gemacht werden?

- Software-Uhr mit „halber Kraft“ bezüglich Hardware-Uhr laufen lassen?
  - ➢ Zu pessimistisch!
  - ➢ Zu ungenau, unpräzise!

Wie weiter?



Wunsch wäre Funktion, die sich immer mehr r(t) nähert, diese Gerade aber nie schneidet

- Erster Zeitstempel als Startwert
- Steigung darf maximal 1 sein!
- Aktuellere Zeitstempel als derzeitige Software-Uhr wären immer genauer

Local Selection Algorithm

LSA ohne Driftkompensation:

Schreibweise:

- Zeitstempel i hat Sendezeit s_i und Empfangszeit r_i
- Empfang eines Zeitstempels löst Algorithmus aus
- Algorithmus bestimmt neue Software-Uhr $c_i(t)$

Algorithmus:

$$c_1 = s_1 \qquad c_i = \max(s_i, c_{i-1}(r_i))$$

$$c_i(t) = c_i + \frac{1}{1+\rho_{\max}^h} \cdot (h(t) - h(r_i)) \qquad (\forall t \geq r_i)$$

Local Selection Algorithm

Erfüllt Funktion der Software-Uhr die Forderungen?

$$c_1 = s_1 \qquad c_i = \max(s_i, c_{i-1}(r_i))$$

$$c_i(t) = c_i + \frac{1}{1+\rho^h_{\max}} \cdot (h(t) - h(r_i))$$

- Werden „aktuellere“ Zeitstempel berücksichtigen?
  - ➢ Wird erreicht durch: $c_i = \max(s_i, c_{i-1}(r_i))$
- Steigung maximal gleich 1?

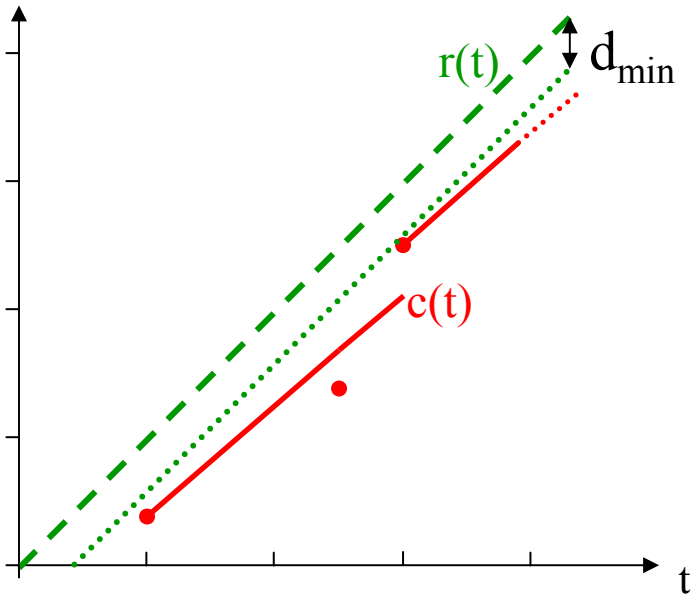
$$\frac{\partial c_i(t)}{\partial t} = \frac{1}{1+\rho^h_{\max}} \cdot \frac{\partial h(t)}{\partial t} < \frac{1}{1+\rho^h_{\max}} \cdot (1+\rho^h_{\max}) = 1$$

Local Selection Algorithm

Was hat dieser Algorithmus für Eigenschaften?

- Verzögerung ist variabel und hat keine normierte Verteilung
  - ➢ Keine mathematische Analyse möglich
- Es braucht also andere qualitative Merkmale
  - ➢ Was ist mit „Safety“ oder „Liveness“ Bedingung?

Eigenschaften von LSA



- Software-Uhr stellt untere Schranke für Referenzzeit dar
- Nur „bessere“ Zeitstempel werden berücksichtigt
  - ➢ Synchronisationsschritt verschlechtert nie die Präzision und verpasst nie die Chance die Präzision zu verbessern

- Uhrendrift wird nicht korrigiert
  - ➢ Erweiterter Algorithmus (LSDC)
- Präzision ist abhängig von der Konstanz des minimalen Delay d_{min}
  - ➢ In Simulation wurde Präzision von 10µs erreicht

Abschliessende Betrachtung

Wird Anforderungskatalog erfüllt?

- ? Energiesparend
- ? Eigenheiten der Netzwerk-Topologie werden berücksichtigt
- ✓ Genügende Präzision
- ✓ Skalierbarkeit
- ✓ robust, z.B. gegen hohe Netzlast, Nachrichtenverlust
- ✓ Kosten und Grösse der Knoten bleiben unverändert
- ✓ Uhren werden nie zurückgestellt
- ✓ Uhrendrift wird korrigiert

Gliederung

1. Motivation & Einführung
2. Anforderungskatalog
3. Verfahren in traditionellen verteilten Systemen
4. Packet Stream Synchronization
5. Reference Broadcasts Synchronization
6. Schlussbetrachtung

RBS

Time Synchronization using Reference Broadcasts

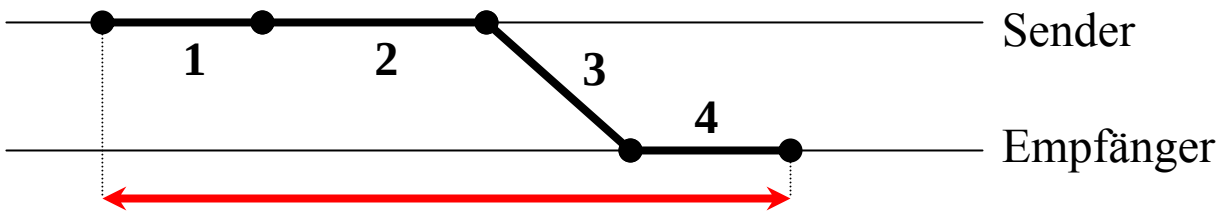
Jeremy Elson, Lewis Girod, Deborah Estrin
Department of Computer Science, University of California

Grundideen:

- Kritischer Pfad wird durch einen Broadcast auf physischer Ebene verkürzt
- Broadcast-Nachricht synchronisiert Empfänger untereinander (nicht mit Sender!)

Der kritische Pfad

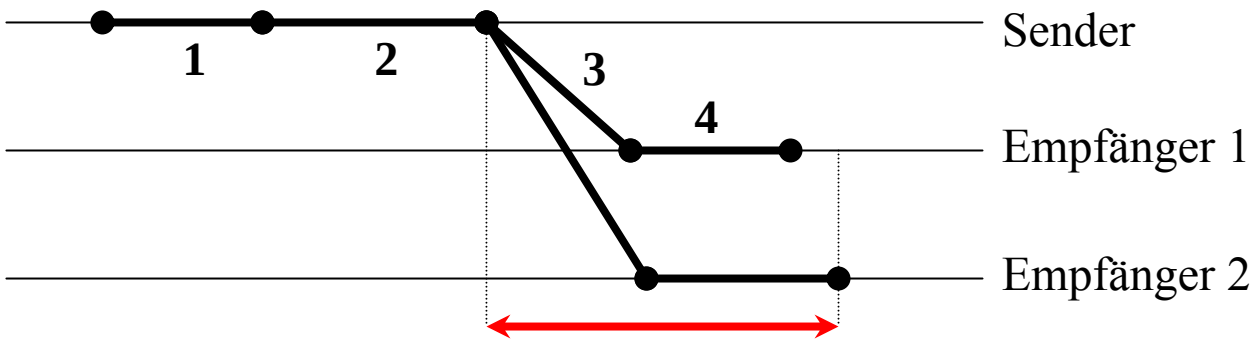
Verzögerung setzt sich zusammen aus:



1. Sendedauer: Konstruieren der Nachricht
2. Zugriffszeit auf Netzwerk: Warten bis gemeinsames Medium frei
3. Flugzeit vom Sender zum Empfänger
4. Empfangsdauer: Entgegennahme der Nachricht

Der kritische Pfad

Broadcast verkürzt diesen kritischen Pfad:



- 1 und 2 für alle Empfänger gleich
- Unterschiede in 3 vernachlässigbar
  - ➢ Ausbreitungsgeschwindigkeit c (10m in 30ns)

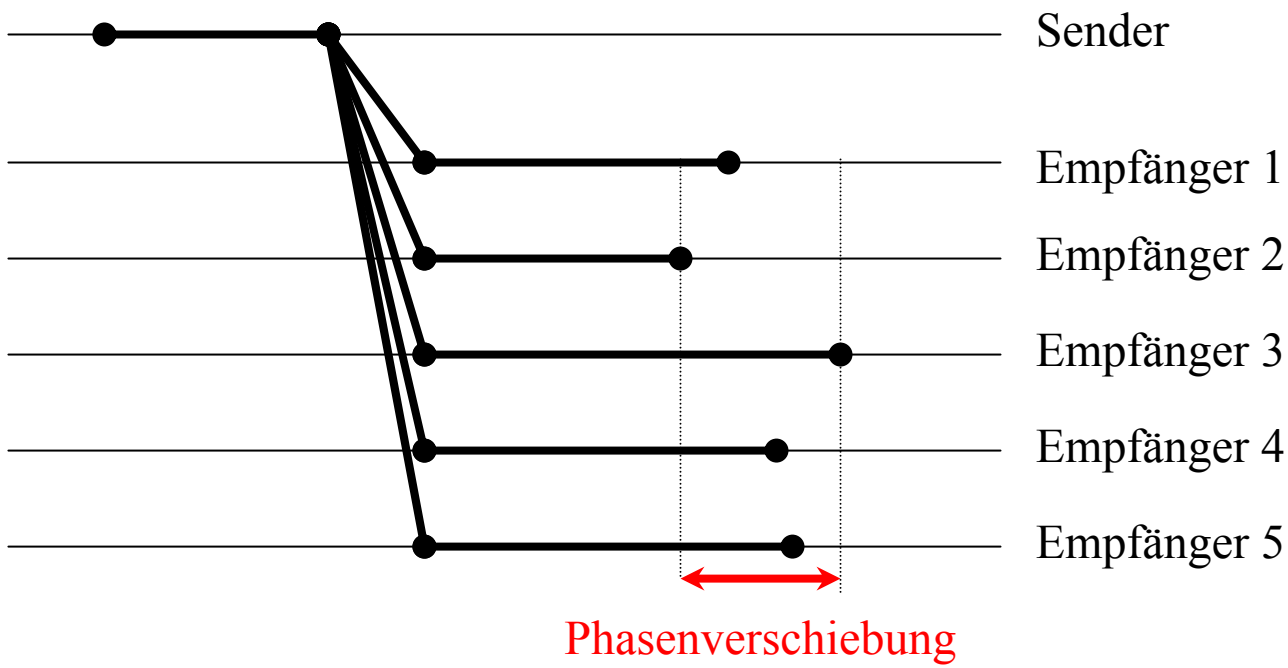
Der kritische Pfad

Einzig die Empfangsdauer bleibt weiterhin nicht-deterministisch:

- Diese kann aber stark verkürzt werden:
  - Ankunft des ersten Bits löst Interrupt aus
  - Lokale Systemzeit dann auslesen
- „Länge“ eines Bits gibt Anhaltspunkt für die Empfangsdauer
  - z.B. bei Berkeley Motes mit 19‘200baud: Theoretisch ca. 52µs, gemessen max. 53.4µs

Synchronisation der Empfänger

2. Grundidee: Es werden nur die Empfänger untereinander synchronisiert!



Synchronisation der Empfänger

Analyse der Phasenverschiebung:

- Phasenverschiebungen zwischen zwei Knoten sind normalverteilt mit Mittelwert $\mu = 0$
  - Berkeley Motes: $\mu = 0$, $\sigma = 11.1\mu s$
- Es darf also angenommen werden, dass die Broadcast-Nachrichten gleichzeitig bei allen Empfängern ankommen.
- Fehler ist dadurch durchschnittlich σ

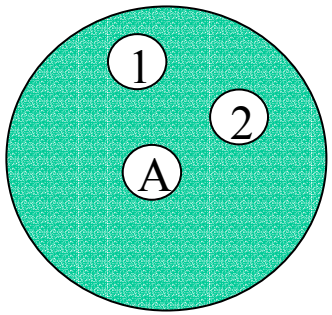
Grundalgorithmus

Einfacher Basisalgorithmus:

- Sender verschickt Broadcast-Nachricht an Empfänger
- Diese zeichnen Empfangszeitpunkt (lokale Uhr) auf
- Empfänger tauschen ihre Ergebnisse unter-einander aus
  - ➢ Empfänger kennen nun den Offset zwischen ihrer Zeit und derjenigen der Nachbarn

Grundalgorithmus

Einfaches Beispiel:



- Knoten 1 hat lokale Uhr c_1
- Knoten 2 hat lokale Uhr c_2
- Knoten A schickt Broadcast-Nachricht an Knoten 1 und Knoten 2

- Knoten 1 erhält Nachricht zur Zeit $t_{c1} = 14$
- Knoten 2 erhält Nachricht zur Zeit $t_{c2} = 17$
- Knoten 1 und Knoten 2 tauschen diese Zeiten aus
- Beide Knoten wissen nun dass: $t_{c1} = t_{c2} - 3$

Grundalgorithmus

Analyse des Grundalgorithmus:

- Algorithmus ist noch zu unpräzise
  - ➢ Verbesserung: Mehrere Broadcast-Nachrichten, durchschnittlicher Offset verwenden
    - ➢ Mit 30 Nachrichten: von 11µs auf 1.6µs
- Uhrendrift wird nicht korrigiert
  - ➢ Informationen über Offsets haben nur kurze Gültigkeitsdauer (Uhrendrift: ca. 1µs/s!)

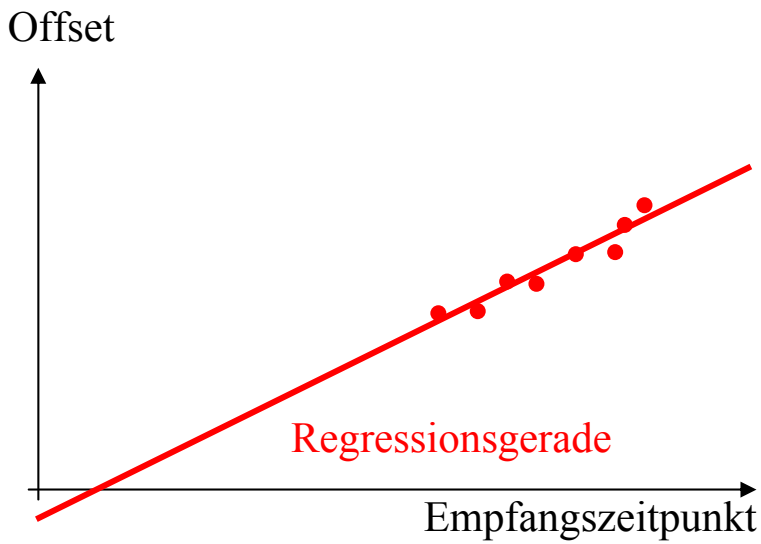
Der RBS Algorithmus

Korrektur des Uhrendrifts durch lineare Regression:

- Mehrere Broadcast-Nachrichten verwenden
- Anstatt Durchschnitt: Gerade durch die Punkte (Empfangszeitpunkt / Offset) legen
  - ➢ Annahme: Uhrendrift nahezu konstant
- Als Resultat dieser Regression erhält man eine Funktion einer Gerade

Der RBS Algorithmus

Jeder Zeitwert lässt sich so konvertieren



- Auch aus der Vergangenheit!
- Post-Facto Synchronisation möglich!
- Synchronisation erst bei Bedarf oder wenn Knoten wieder erreichbar ist

- Damit lässt sich Energie sparen!

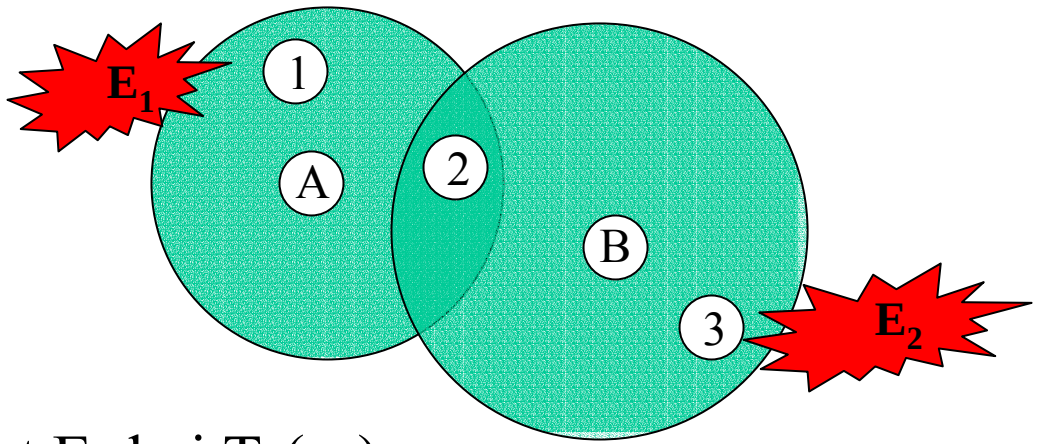
Vergleich mit NTP

Test mit IPAQs und IEEE 802.11 WLAN:

Netzlast		μ	σ
schwach	RBS	6.29μs	6.45μs
	NTP	51.18μs	53.30μs
stark (6MBit/s)	RBS	8.44μs	9.37μs
	NTP	1542.27μs	1192.53μs

- RBS über 8 bis 180 mal besser als NTP!
- RBS relativ stabil bei starker Netzlast!
- RBS mit Interrupt-Zeitstempel : 1.85μs!

Multi-Hop Synchronisation



Beispiel:

- n_1 beobachtet E_1 bei $T_1(n_1)$
- n_3 beobachtet E_2 bei $T_2(n_3)$
- n_2 kann dank RBS (von A) $T_1(n_1)$ zu $T_1(n_2)$ konvertieren
- n_3 kann dank RBS (von B) $T_1(n_2)$ zu $T_1(n_3)$ konvertieren
- n_3 kann nun $T_1(n_3)$ und $T_2(n_3)$ vergleichen!

Multi-Hop Synchronisation

Mit RBS über mehrere Hops synchronisieren:

- Grundidee: Zeit bei jedem Hop konvertieren
- Jede Umwandlung erhöht aber den Fehler!
  - ➢ Durchschnittliche Abweichung nach n Hops:

$$\sigma_{Total} = \sigma_1 \sqrt{n}$$

wegen Var(x+y) = Var(x) + Var(y)

und Var = σ^2

Einige Bemerkungen

- RBS braucht ein broadcastfähiges Netz!
- Beliebiger Knoten kann Sender sein
  - Braucht keine statische Struktur
- Kein Zeitstempel in Broadcast-Nachricht nötig!
- Synchronisation mit einer externen Zeit (z.B. UTC) ist einfach möglich
  - Spezieller Knoten einfügen

Abschliessende Betrachtung

Erfüllt RBS den Anforderungskatalog?

- ✓ Energiesparend
- ✓ Eigenheiten der Netzwerk-Topologie werden berücksichtigt
- ✓ Sehr hohe Präzision
- ✓ robust, z.B. gegen hohe Netzlast, Nachrichtenverlust
- ✓ Kosten und Grösse der Knoten bleiben unverändert
- ✓ Zeit bleibt kontinuierlich, Uhren werden nicht verändert
- ✓ Uhrendrift wird korrigiert

- ➢ Mangel: Skalierbarkeit?

Gliederung

1. Motivation & Einführung
2. Anforderungskatalog
3. Verfahren in traditionellen verteilten Systemen
4. Packet Stream Synchronization
5. Reference Broadcasts Synchronization
6. Schlussbetrachtung

Schlussbetrachtung

- Es existieren verschiedene Ansätze für das Problem der Zeitsynchronisation
  - Unterschiedliche Voraussetzungen
  - Unterschiedliche Eigenheiten

➢ Dies soll auch so sein!

➢ Unterschiedliche Anwendungen haben unterschiedlichen Anforderungen

Schlussbetrachtung

- Es kann keinen Algorithmus geben, der für alle Anwendungen und Gegebenheiten optimal ist!
- Präzision ⟷ Energie
- RBS scheint ein guter Algorithmus zu sein
  - Erfüllt Anforderungskatalog fast
  - Setzt allerdings broadcastfähiges Kommunikationsmedium voraus!

Fazit

Der Aspekt der Zeitsynchronisation sollte schon am Anfang beim Hardware-Design des Sensornetzes berücksichtigt werden