

Zeit

- Motivation

- Konvergenz-Algorithmus CNV

- Network Time Protocol (NTP)

- Logische Uhren



- Zeit als Mittel zur Reihenfolgebestimmung (Beispiele)
 - Erkennung von Modifikationen an Dateien (z. B. bei `make`)
 - Protokollierung von Ereignissen zu Debugging-Zwecken
- Problem: Völlig identische physikalische Uhren existieren nicht
 - Unterschiedliche Offsets bei der Initialisierung
 - Abweichende Ganggeschwindigkeiten (Frequenzfehler)
 - Umgebungseinflüsse (z. B. Bauteilalterung, Temperaturschwankungen)
- Beobachtungen in Bezug auf verteilte Systeme
 - Regelmäßige Synchronisierung von Uhren erforderlich
 - Physikalische Zeitstempel für manche Anwendungsfälle zu grobgranular
- Herausforderungen
 - Wie lassen sich physikalische Uhren möglichst präzise synchronisieren?
 - Wie können Ereignisse ohne physikalische Zeitstempel geordnet werden?



■ Problemstellung

- Uhrensynchronisation trotz fehlerhafter Protokollteilnehmer
- Fehlermodell
 - Bis zu f der insgesamt n Rechner können *byzantinisch* fehlerhaft sein
 - Fehlerhafte Rechner senden eventuell (absichtlich) falsche Zeitwerte
- Abweichung der Uhren korrekter Rechner soll höchstens δ betragen

■ Konvergenz-Algorithmus CNV

- Periodische Kombination der Uhrzeiten verschiedener Rechner
- Bestimmung der Uhrzeit eines entfernten Rechners
 - Abschätzung der Differenz zwischen der lokalen und der entfernten Uhr
 - Berechnung des absoluten Werts bei Bedarf: Aktuelle lokale Zeit + Differenz
- Für Korrektheit erforderliche Anzahl an Teilnehmern: $n > 3f$

■ Literatur



Leslie Lamport and P. M. Melliar-Smith
Synchronizing clocks in the presence of faults
Journal of the ACM, 32(1):52–78, 1985.



■ Variablen

- Z_r Lokale Uhrzeit des Rechners r
- $Z_{r,s}$ Sicht von Rechner r auf die aktuelle Uhrzeit von Rechner s

■ Annahmen

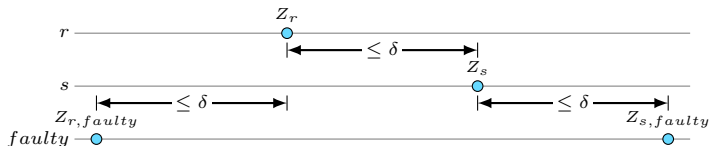
- Initial: Die Uhrzeiten aller Rechner unterscheiden sich um höchstens δ
- Vernachlässigbar
 - Ausführungsdauer des Algorithmus
 - Taktratenunterschiede zwischen den Uhren korrekter Rechner
 - Ungenauigkeit beim Auslesen der Uhrzeit eines korrekten Rechners

■ Algorithmus für einen Rechner r

1. Aktualisierung der Sicht auf die Uhrzeiten $Z_{r,i}$ jedes anderen Rechners i
2. Identifizierung und Unterdrückung von Ausreißern
 - Kategorisierung als Ausreißer, falls $|Z_r - Z_{r,i}| > \delta$
 - Ersetzen durch lokale Uhrzeit: $Z_{r,i} := Z_r$
3. Aktualisierung der eigenen Uhrzeit durch Mittelwertbildung: $Z_r := \frac{\sum_i Z_{r,i}}{n}$



- Abweichungen zwischen den Sichten korrekter Rechner r und s
 - Sicht auf die Uhr eines korrekten Rechners: $Z_{r,correct} \approx Z_{s,correct}$
 - Sicht auf die Uhr eines fehlerhaften Rechners: $|Z_{r,faulty} - Z_{s,faulty}| \leq 3\delta$



- Maximale Abweichung der lokalen Uhrzeiten nach der Aktualisierung

$$\begin{aligned} |Z_r - Z_s| &= \left| \frac{(n-f) \cdot Z_{r,correct} + f \cdot Z_{r,faulty}}{n} - \frac{(n-f) \cdot Z_{s,correct} + f \cdot Z_{s,faulty}}{n} \right| \\ &\approx \left| \frac{f \cdot Z_{r,faulty}}{n} - \frac{f \cdot Z_{s,faulty}}{n} \right| = \frac{f}{n} \cdot |Z_{r,faulty} - Z_{s,faulty}| \\ &\leq \frac{f}{n} \cdot 3\delta = \frac{3f}{n} \cdot \delta \end{aligned}$$

\Rightarrow Für $n > 3f$: $|Z_r - Z_s| < \delta$



Network Time Protocol (NTP)

■ Network Time Protocol (NTP)

■ Genauigkeit

- Lokales Netz < 1 ms
- Weitverteiltes Netz ~ 10 ms

■ Implementierung

- Einsatz von 64-Bit-Zeitstempeln
- Kommunikation per UDP

■ Zusammenschluss von Referenz-Servern auf mehreren Hierarchiestufen (*Strata*)

- Stratum 0 Zeitgeber (z. B. Atomuhren)
- Stratum 1 Primäre NTP-Server
- Stratum $i > 1$ Abhängige NTP-Server

■ Fehlertoleranz durch Interaktion mit mehreren Referenz-Servern

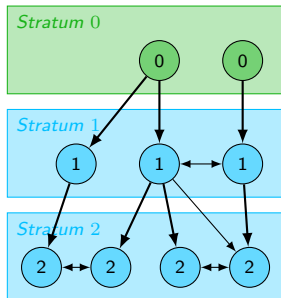
■ Literatur



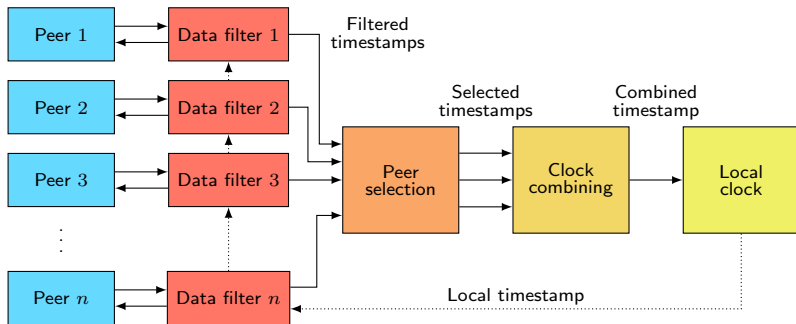
David L. Mills

Internet time synchronization: The network time protocol

IEEE Transactions on Communications, 39(10):1482–1493, 1991.

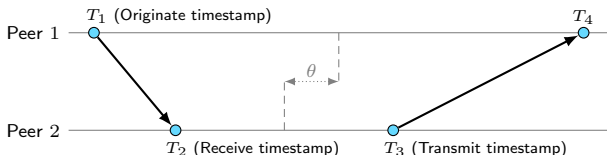


- Austausch von Zeitstempeln mit mehreren Referenz-Servern (*Peers*)
- Verarbeitung von Zeitstempeln
 - Bestimmung eines Referenzzeitstempels pro Peer durch Filterung
 - Auswahl (vermeintlich) präziser Peers
 - Kombination der selektierten Informationen
- Aktualisierung des Regelmechanismus der lokalen Uhr



■ Durchführung von Messungen

- Weitergabe von Zeitstempeln per Nachrichtenaustausch zwischen Peers
- Bestimmung der Nachrichtenlaufzeit $\delta = (T_4 - T_1) - (T_3 - T_2)$
- Abschätzung der Uhrenabweichung
 - Offset zwischen zwei Uhren: $\theta = \frac{T_2 + T_3}{2} - \frac{T_1 + T_4}{2}$
 - Exakter Wert, falls Laufzeiten in beide Richtungen identisch
 - Maximaler Fehler bei asymmetrischen Laufzeiten: $\frac{\delta}{2}$



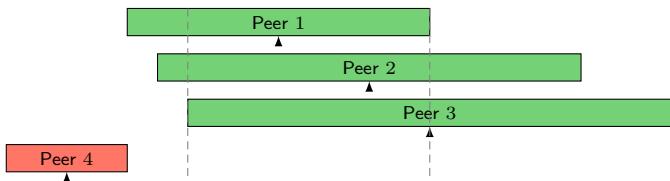
■ Filterung von Messwerten für jeden Peer

- Betrachtung der letzten 8 Wertpaare (δ, θ)
- Bevorzugung von Messergebnissen mit kürzeren Nachrichtenlaufzeiten
- Benachteiligung älterer Werte bei Abschätzung von Messfehlern



■ Auswahl präziser Peers


- Trennung genauer Knoten („*truechimers*“) von ungenauen („*falsetickers*“)
- Berücksichtigung von Messfehlern durch Einsatz von Konfidenzintervallen
- Suche nach einem Intervall X mit folgenden Eigenschaften
 - X ist vollständig in jedem Konfidenzintervall genauer Knoten enthalten
 - X enthält alle Mittelpunkte der Konfidenzintervalle genauer Knoten
- Abbruch, falls weniger als die Hälfte der Knoten als „genau“ eingestuft



■ Kombination der ausgewählten Zeitstempel

- Bevorzugung von Peers mit kleinem Stratum
- Berechnung eines gewichteten Mittelwerts der Offsets selektierter Peers



- Problemstellung
 - Erstellung einer Ordnung auf Ereignisse in einem verteilten System
 - Annahme: Physikalische Zeitstempel zu ungenau
- Lösungsansatz: Einsatz von *logischen Uhren*
 - Einführung einer „ereignete sich vor“-Relation „ \rightarrow “ („*happened before*“)
 - Bedingungen für verschiedene Ereignisse a , b und c
 - Falls sich a auf demselben Knoten wie b und vor b ereignete, dann $a \rightarrow b$
 - Falls a das Senden einer Nachricht ist und b ihr Empfang, dann $a \rightarrow b$
 - Falls $a \rightarrow b$ und $b \rightarrow c$ gilt, dann muss auch $a \rightarrow c$ gelten
 - Ereignisse a und b sind *nebenläufig*, falls $a \nrightarrow b$ und $b \nrightarrow a$ gilt
 - Praktische Umsetzung in Form von *Lamport-Uhren*
- Literatur
 -  Leslie Lamport
Time, clocks, and the ordering of events in a distributed system
Communications of the ACM, 21(7):558–565, 1978.

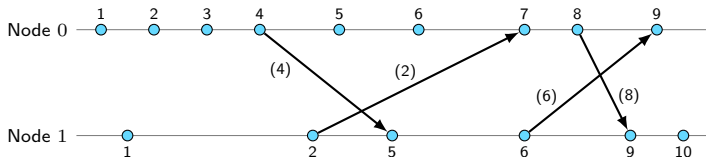


■ Annahmen

- Jeder Knoten i im System verfügt über einen Zähler C_i („Uhr“)
- Relevante Ereignisse: Versand/Empfang von Nachrichten, lokale Aktionen

■ Algorithmus

- Lokale Aktionen führen jeweils zur Erhöhung des Zählers um 1
- Ereignis s : Versand einer Nachricht durch Knoten i
 1. Erhöhung des Zählers $C_i := C_i + 1$
 2. Hinzufügen eines Sendezeitstempels $C\langle s \rangle := C_i$ zur Nachricht
- Ereignis e : Empfang einer Nachricht mit Zeitstempel $C\langle s \rangle$ auf Knoten j
 1. Ermittlung eines Empfangszeitstempels $C\langle e \rangle := \max(C_j, C\langle s \rangle) + 1$
 2. Setzen der lokalen Uhr auf $C_j := C\langle e \rangle$



■ Eigenschaften



- Erzeugung einer **partiellen** Ordnung auf der Menge aller Ereignisse
- Existenz von „gleichzeitigen“ Ereignissen möglich
- Zeitstempel (potentiell) kausal abhängiger Ereignisse
 - Annahme: Ereignis a hat Ereignis b beeinflusst
 - Folge: $C\langle a \rangle < C\langle b \rangle$
- Kein Umkehrschluss von Zeitstempeln auf kausale Abhängigkeit möglich
 - Annahme: Für zwei Zeitstempel $C\langle c \rangle$ und $C\langle d \rangle$ gilt $C\langle c \rangle < C\langle d \rangle$
 - Ereignis d kann von c (potentiell) beeinflusst worden sein oder auch nicht

■ Erstellung einer **totalen** Ordnung

- Vergabe einer eindeutigen ID i für jeden beteiligten Knoten
- Zeitstempel (C_i, i) : Kombination aus lokaler Zeit und Knoten-ID
- Anordnung: $(C_i, i) < (C_j, j) \Leftrightarrow C_i < C_j \vee (C_i = C_j \wedge i < j)$
- Anwendungsbeispiel: *Lamport-Locks*

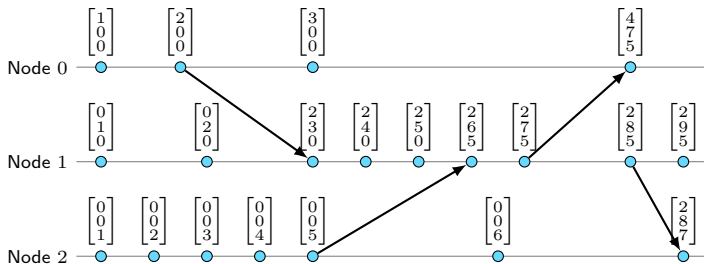
[Siehe 6. Übungsaufgabe]



- Problem bei Lamport-Uhren
 - Nutzung derselben Zeitlinie durch alle beteiligten Knoten
 - Zeitstempel lassen keine Rückschlüsse auf mögliche Zusammenhänge zu
- Vektoruhren
 - Erweiterung des Lamport-Uhren-Prinzips
 - Verwaltung einer eigenen Zeitlinie für jeden beteiligten Knoten
- Literatur
 -  Colin J. Fidge
Timestamps in message-passing systems that preserve the partial ordering
Proceedings of the 11th Australian Computer Science Conference (ACSC '88), S. 55–66, 1988.
 -  Friedemann Mattern
Virtual time and global states of distributed systems
Parallel and Distributed Algorithms, 1(23):215–226, 1989.



- Annahmen
 - N ist die Anzahl der Knoten im System
 - Jeder Knoten i verfügt über einen Zähler-Vektor \vec{C}_i der Länge N
- Hauptunterschiede zu Lamport-Uhren
 - Ereignisse auf Knoten i führen zur Erhöhung des i -ten Zählers $\vec{C}_i[i]$
 - Komponentenweise Kombination von Zeitstempeln bei Empfang von $\vec{C}\langle s \rangle$
 - $\vec{C}_i[i] := \vec{C}_i[i] + 1$
 - $\vec{C}_i[x] := \max(\vec{C}_i[x], \vec{C}\langle s \rangle[x])$ für $0 \leq x \neq i < N$



- Vergleich von Vektoruhren
 - Einführung einer „ist kleiner als“-Relation „ \prec “ für Vektoruhren
 - $\vec{C}_i \prec \vec{C}_j \Leftrightarrow (\forall x : \vec{C}_i[x] \leq \vec{C}_j[x]) \wedge (\exists x : \vec{C}_i[x] < \vec{C}_j[x])$
- Identifizierung (potentiell) kausal abhängiger Ereignisse möglich
 - $\vec{C}\langle a \rangle \prec \vec{C}\langle b \rangle$ Ereignis b wurde eventuell von Ereignis a beeinflusst
 - $\vec{C}\langle a \rangle \not\prec \vec{C}\langle b \rangle$ Ereignisse a und b sind unabhängig voneinander
- Bestimmung der **kausalen Vergangenheit** eines **Ereignisses**

