

10 Zeit in verteilten Systemen

- 10.1 Überblick
- 10.2 Uhrensynchronisation
- 10.3 Logische Uhren
- 10.4 Synchr. von physikal. Uhren



- Uhrensynchronisation
- Zeit im Verteilten System
- Logische Uhren
- Synchronisation von physikalischen Uhren
 - Konvergenz-Algorithmus - CNV
 - Network Time Protocol - NTP



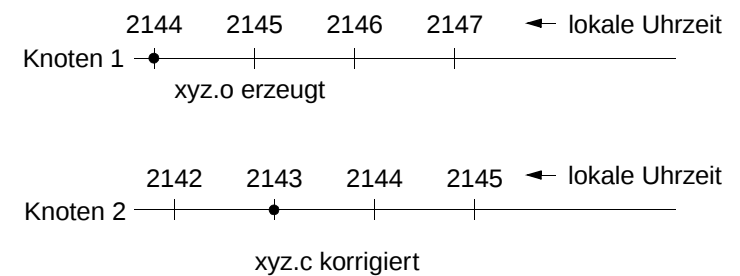
Uhrensynchronisation

- Notwendigkeit von Uhrensynchronisation
 - Zeit als Mittel der Reihenfolgebestimmung
- Probleme der Uhrensynchronisation
- Logische Uhren
 - Lamport-Uhren
 - Vektoruhren
- Synchronisation von physikalischen Uhren
 - Grundlagen
 - Konvergenzalgorithmus CNV
 - Network Time Protocol NTP



Zeit als Mittel der Reihenfolgebestimmung

- Beispiel: make

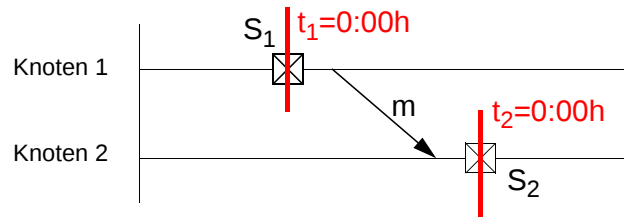


- ⇒ Modifikation von `xyz.c` durch Knoten 2 wird nicht erkannt
- ⇒ Die Datei wird nicht neu übersetzt!



Zeit als Mittel der Reihenfolgebestimmung

- Beispiel: Verteilte Zustandssicherung
 - Gegeben: Verteiltes System aus interagierenden Knoten
 - Für eine konsistente Zustandsicherung soll bei allen Knoten zu festgelegter Uhrzeit der Zustand gesichert werden
 - Problem: Inkonsistente Sicherungspunkte möglich, falls die lokalen Uhren voneinander abweichen:



⇒ Auswirkung der Nachricht m in S_2 enthalten, aber nicht in S_1 !



Probleme der Uhrensynchronisation

- Es gibt keine völlig identischen physikalische Uhren
 - Abweichende Initialisierung (konstantes Offset)
 - Abweichende Ganggeschwindigkeit (Frequenzfehler)
 - Umgebungseinflüsse (z.B. Bauteilalterung, **Temperaturabhängigkeit**)⇒ Ohne Synchronisierung kann Fehler immer größer werden!
- Gemeinsame Uhr für alle Knoten eines verteilten Systems (meist) nicht realisierbar
- „Zentrale Referenzuhren“ (Funk, z.B. Langwellensender DCF77 in Mainflingen, amerikanischer Kurzwellensender WWV in Fort Collins, GPS) nur mit technisch beschränkter Genauigkeit



Logische Uhren

- Grundidee (Logische Uhren nach Lamport):
Aussagen zur Reihenfolge unterschiedlicher Ereignisse
 - Nur benötigt, wenn eine Interaktion zwischen einzelnen Komponenten des verteilten Systems erfolgt ist!
 - Eine Aktion b , die logisch durch die Interaktion bedingt „nach“ einer Aktion a stattfindet, soll stets den größeren Zeitstempel tragen⇒ Synchronisation bei Kommunikation
- Anmerkung: Die im Folgenden beschriebenen Verfahren setzen voraus, dass die Kommunikation nur durch Nachrichtenaustausch über ein Kommunikationsnetz erfolgt

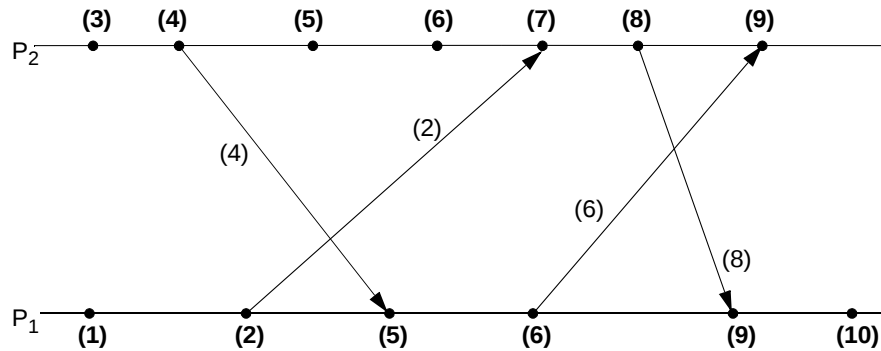


Logische Uhren

- Beschreibung des Algorithmus: Jeder Prozess i verfügt über einen Zähler C_i („Uhr“), der auf folgende Weise zählt:
 - Bei Ausführung einer lokalen Aktion und bei Ausführung einer Sende-Aktion durch Prozess i wird seine Uhr C_i um 1 erhöht; die Aktion bekommt den Wert nach dem Erhöhen als Zeitstempel
 - Jede Nachricht m trägt als Zeitstempel t_m den Zeitstempel der Sendeaktion
 - Bei Empfang einer Nachricht durch Prozess i bei lokalem Uhrenstand C_i und empfangenem Zeitstempel t_m wird die lokale Uhr auf $\max(C_i, t_m) + 1$ gestellt und dieser Wert als Zeitstempel des Empfangsereignis verwendet



- Beispiel für zwei Prozesse P_1 und P_2



- Eigenschaften

- Potentielle kausale Abhängigkeit eines Ereignisses E_2 von einem Ereignis E_1 [Kurznotation: $E_1 \rightarrow E_2$]:

$$E_1 \rightarrow E_2 \Rightarrow t(E_1) < t(E_2)$$

- Die Umkehrung

$$t(E_1) < t(E_2) \Rightarrow E_1 \rightarrow E_2$$

gilt nicht!

- Die Zeitstempel erzeugen eine **partielle** Ordnung auf der Menge aller Ereignisse. (Es gibt Ereignisse die „gleichzeitig“ auftreten und somit nicht geordnet werden können.)

- Ergänzung zu vollständiger Ordnung

- Jeder Prozess erhält eindeutige Identifikation
- Zeitstempel bestehend aus Prozess i und lokaler Zeit C_i .
- Anordnung:

$$(C_i, i) < (C_k, k) \Leftrightarrow C_i < C_k \text{ oder } (C_i = C_k \text{ und } i < k)$$

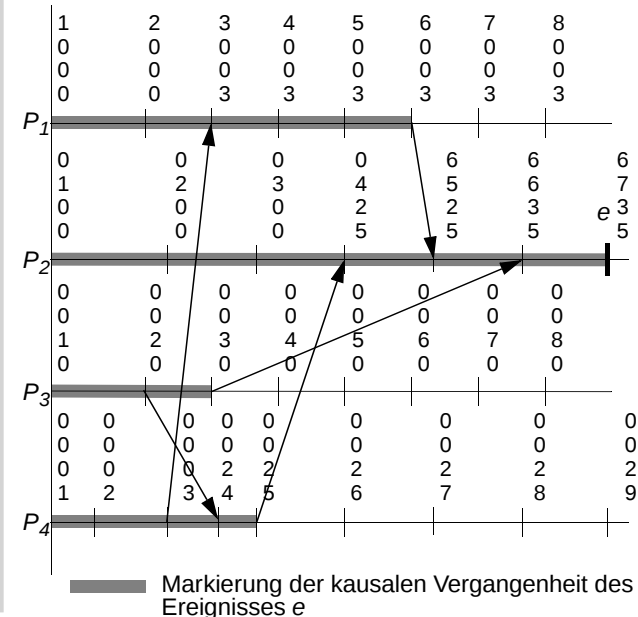
- Jeder Knoten besitzt nun eine lokale Uhr, die aus einem Vektor der Länge N (= Anzahl der vorhandenen Knoten) besteht

- Implementierung:

1. Initialisierung jedes Zeitvektors mit dem Nullvektor.
2. Bei jedem lokalen Ereignis eines Prozesses P_i wird dessen Komponente in seinem Zeitvektor inkrementiert: $C_i[i]++$
3. Ein Prozess P_i , der eine Nachricht empfängt, inkrementiert seine Komponente im Zeitvektor und kombiniert diesen danach komponentenweise mit dem empfangenen Zeitvektor t :

$$C_i[i]++$$

$$\text{für } k = 1 \dots N: C_i[k] := \max(C_i[k]; t[k])$$



Markierung der kausalen Vergangenheit des Ereignisses e

Logische Uhren

- Eigenschaften
 - Erzeugt eine kausale Ordnung: $t(E_1) < t(E_2) \Leftrightarrow E_1 \rightarrow E_2$
Definition von $<$ mit $t(E_1) := (a_1, \dots, a_n)$ und $t(E_2) := (b_1, \dots, b_n)$:
$$t(E_1) < t(E_2) \Leftrightarrow (\forall i : a_i \leq b_i) \wedge (\exists i : a_i < b_i)$$
- Vorteil
 - Exakte Aussage zum kausalen Zusammenhang von Ereignissen mit Hilfe des Zeitstempels möglich
- Nachteil
 - Hoher Kommunikationsaufwand (Vektor aus N Elementen in jeder Nachricht; skaliert schlecht für großes N !)



Synchronisation von physikalischen Uhren

- Physikalische Zeit basierend auf Atom-Sekunde: TAI
 - „Die Sekunde ist das 9.192.631.770-fache der Periodendauer der dem Übergang zwischen den beiden Hyperfeinstrukturniveaus des Grundzustandes von Atomen des Nuklids ^{133}Cs entsprechenden Strahlung“
[Gültige Definition seit 1967, davor über Erdrotation (bis 1956) bzw. über Erdumlaufzeit um die Sonne festgelegt.]
 - Genauigkeit von Cs-Uhren: bis ca. 10^{-14} (entspricht ca. $0.8\text{ns}/\text{Tag}$!)
- Verbreitung der amtlichen Zeit: Normalfrequenz/Zeitzeichensender
 - Langwelle: z.B. DCF77 (Mainflingen, D); WWVB (Boulder, US); maximale Genauigkeit bis ca. $50\mu\text{s}$
 - Kurzwelle: z.B. WWV (Colorado), WWVH (Hawaii); maximale Genauigkeit bis ca. 1ms
 - GPS-Satelliten: maximale Genauigkeit ca. $0.3\mu\text{s}$



Synchronisation von physikalischen Uhren

- Softwarebasierte Synchronisation
 - Master-/Slave mit Verteilung einer Referenzzeit (broadcast oder poll)
 - Einigungsverfahren
- Charakterisierungsmöglichkeiten von Algorithmen
 - Monotonie (Zeitsprünge bei Korrekturen?)
 - Synchronisation mit der amtlichen Zeit
 - Robustheit gegen Netzpartitionierung
 - Fehlertoleranz gegen falsch gehende Referenzuhren
 - Referenztreue
 - Erzeugte Netzlast
 - Fehleraussage



Konvergenz-Algorithmus CNV

- Aufgabe des Algorithmus
 - Vermeidung einer Akkumulation einer immer größer werdenden Abweichung zwischen mehreren Uhren
- Ablauf
 - Jeder Prozess liest die Uhr jedes anderen Prozesses
 - Er stellt seine eigene Uhr auf den Mittelwert, wobei für Uhren, die **mehr** als eine festgelegte Schranke δ von der eigenen abweichen, der Wert der **eigenen** Uhr genommen wird
 - Die gewünschten Eigenschaften (Konvergenz gegen eine gemeinsame Uhrzeit, Tolerierung von fehlerhaften Uhren) werden erfüllt, wenn weniger als ein Drittel der Uhren fehlerhaft ist



Konvergenz-Algorithmus CNV

■ Annahmen

- Benötigt $n > 3m$ Prozesse, um m fehlerhafte Prozesse zu tolerieren
- Initial sind alle Prozesse auf in etwa die gleiche Uhrzeit synchronisiert (Fehler kleiner als δ)
- Die Uhren aller korrekten Prozesse laufen mit in etwa der korrekten Rate
- Ein korrekter Prozess kann die Uhr eines anderen korrekten Prozess mit (vernachlässigbar) kleinem maximalen Fehler ϵ lesen

■ Ziel

- Zu jeder Zeit sollen alle korrekten Prozesse in etwa die gleiche Uhrzeit haben (Fehler kleiner als δ)



Konvergenz-Algorithmus CNV

■ Plausibilitätsbetrachtung

- Definitionen
 - p, q seien fehlerfreie Prozesse, r irgendein Prozess
 - $C_{p,q}$ sei die Uhrzeit von q , so wie sie p bekannt ist
- Es gilt:
 - r fehlerfrei $\Rightarrow C_{p,r} \approx C_{q,r}$
 - r fehlerhaft $\Rightarrow |C_{p,r} - C_{q,r}| < 3\delta$
- Alle Prozesse p stellen ihre Uhr auf $\sum_i (C_{p,i})/n$



Konvergenz-Algorithmus CNV

■ Plausibilitätsbetrachtung

- Schlechtester Fall:

$$\begin{aligned} (n-m)\text{-mal } (C_{p,i} - C_{q,i}) &\approx 0 \\ m\text{-mal } (C_{p,i} - C_{q,i}) &< 3\delta \end{aligned}$$

- Also: neue Differenz zwischen p und q ist $(3m/n) * \delta$
Mit $n > 3m$ folgt: neue Differenz $< \delta$
- Vernachlässigt sind dabei
 - die Zeit für die Ausführung des Algorithmus sowie
 - Fehler durch nicht gleichzeitiges Ablesen der Uhren



Das Network Time Protocol – NTP

■ Ausführliche Informationen zu NTP

- <http://www.ntp.org> (Offizielle NTP-Homepage)
- <http://www.eecis.udel.edu/~mills> (Homepage David Mills)

■ Geschichte

- Entwickelt seit 1982 (NTPv1, RFC 1059) unter Leitung von D. Mills
- Seit 1990 NTPv3 (teilweise immer noch in Verwendung)
- Aktuelle Version NTPv4, seit 1994

■ Eigenschaften von NTP

- Zweck: Synchronisierung von Rechneruhren im bestehenden Internet
- Derzeit weit über 100.000 NTP-Knoten weltweit
 - 151 aktive öffentliche Stratum 1 - Knoten (Stand Okt. 2005)
 - 208 aktive öffentliche Stratum 2 - Knoten
- Erreichbare Genauigkeiten von ca. 10ms in WANs; $< 1ms$ in LANs
- NTP-Dämon auf fast allen Rechnerplattformen verfügbar, von PCs bis Crays; Unix, Windows, OS X, eingebettete Systeme
- Fehlertolerant



Das Network Time Protocol – NTP

■ Grundlegender Überblick

- Primäre Server (*Stratum 1*), über Funk oder Standleitungen an amtliche Zeitstandards angebunden
- Synchronisation weiterer Knoten nach primären Servern über ein selbstorganisierendes, hierarchisches Netz
- Verschiedene Betriebsarten (Master/Slave, symmetrische Synchronisation, Multicast, jeweils mit/ohne Authentisierung)
- Zuverlässigkeit durch redundante Server und Netzpfade
- Optimierte Algorithmen, um Fehler durch Jitter, wechselnde Referenzuhren und fehlerhafte Server zu reduzieren
- Lokale Uhren werden in Zeit und Frequenz durch einen adaptiven Algorithmus geregelt

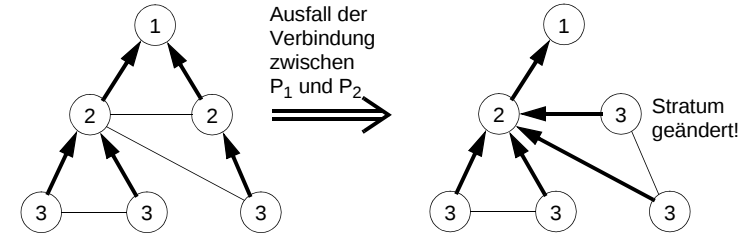


Das Network Time Protocol – NTP

■ Stratum:

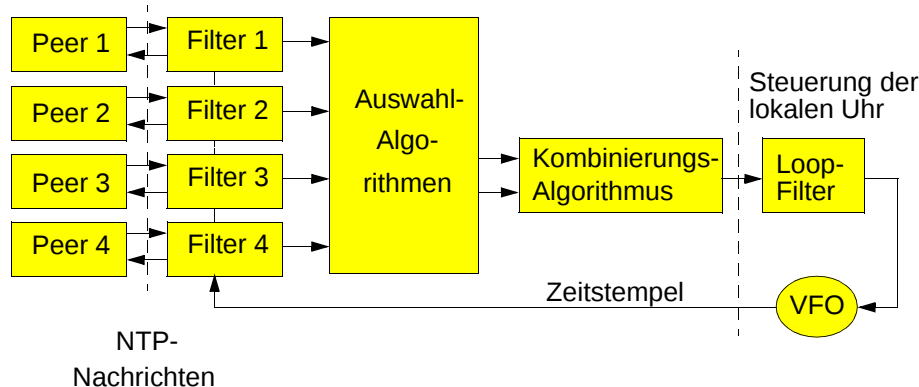
- 1: primärer Zeitgeber
- $i > 1$: synchronisiert mit Zeitgeber des Stratums $i - 1$

■ Stratum kann dynamisch wechseln:



Das Network Time Protocol – NTP

■ Architektur-Überblick



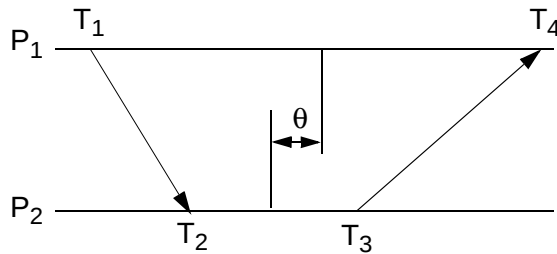
Das Network Time Protocol – NTP

- **Mehrere Referenzserver** („Peer“) für Redundanz und Fehlerstreuung
- **Peer-Filter** wählen pro Referenzserver den jeweils besten Wert aus den **acht** letzten Offset-Messwerten aus
- Die **Auswahlalgorithmen** versuchen zunächst richtig gehende Uhren („*truechimers*“) zu erkennen und falsche Uhren („*falsechimers*“) herauszufiltern, und wählen dann möglichst genaue Referenzuhren aus
- Der **Kombinationsalgorithmus** berechnet einen gewichteten Mittelwert der Offset-Werte (frühere NTP-Versionen wählen einfach den am vertrauenswürdigsten erscheinenden Referenzknoten aus)
- **Loop Filter** und **VFO** (Variable Frequency Oscillator) bilden zusammen die geregelte lokale Uhr. Die Regelung ist so entworfen, dass Jitter und Drift bei der Regelung minimiert werden



■ Offset-Messung

- Es werden die letzten 8 Messungen gespeichert;
Index $i = 0$: neueste Messung

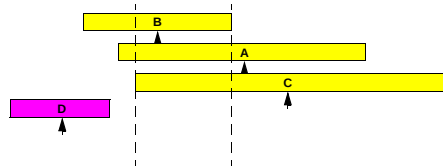


- Reine Nachrichtenlaufzeit: $\delta = (T_4 - T_1) - (T_3 - T_2)$
- Geschätztes Offset: $\theta = (T_2 + T_3)/2 - (T_1 + T_4)/2$
 - Exakter Wert, falls Laufzeiten in beide Richtungen gleich sind
 - Maximaler Fehler bei unsymmetrischen Laufzeiten: $\delta/2$



■ Peer-Filter-Algorithmus

- Getrennte Ausführung pro Peer
- Bei jeder Messung ermittelte Werte:
 - Offset θ , Laufzeit δ
 - Abschätzung des Messfehlers:
 $\epsilon = \text{Lesegenauigkeit} + \text{MAXDRIFT} * (T_4 - T_1)$
- Eingetragene Messwerte werden in einen Puffer eingetragen
 - Speicherung der letzten 8 Messwerte
 - Aktualisierung der Fehlerabschätzung ϵ bei jeder neuen Messung um den möglichen Fehler durch Alterung (Uhrendrift)
 $i = 7 \dots 1 : (\delta_i, \theta_i, \epsilon_i) = (\delta_{i-1}, \theta_{i-1}, \epsilon_{i-1} + \text{MAXDRIFT} * \text{verstrichene Zeit})$
 $i = 0 : (\delta_0, \theta_0, \epsilon_0) = \text{neuer Messwert}$
- Weitere Verarbeitung der Messwerte und Ermittlung eines *Korrektheitsintervalls* für den Peer



■ Auswahl-Algorithmus

- Trennung von „truechimers“ und „falsetickers“
- DTS (Digital Time Service, Vorgänger-Algorithmus, einfacher):
Ermittle Durchschnitt mit den meisten überlappenden Korrekturintervallen. Mittelpunkt des Intervalls wird als Offset zur Uhrenkorrektur verwendet.
- Ziel bei NTP: Auswahl des Intervalls so, dass die Mittelpunkte der Intervalle der als korrekt betrachteten Knoten im Intervall liegen



■ Kombinationsalgorithmus

- Weiteres Aussortieren vergleichsweise schlechter Referenzen, bevorzugte Selektion von Referenzen mit kleinem Stratum
→ Zusammenstellung einer nach Qualität sortierten Liste von Referenzen (beste Referenz steht vorne)
- Die übrig bleibenden Knoten werden zur Synchronisation verwendet
 - Einfache Variante: Falls der bisherige Referenzknoten sich in der Liste befindet, wird dieser weiterhin zur Synchronisation verwendet; ansonsten wird der Knoten am Anfang der Liste zum neuen Synchronisationsknoten
 - Komplexe Variante (NTPv4): Berechnung eines gewichteten Mittelwerts der Offsets aus allen Knoten
- Das so ermittelte Offset wird an die Uhrenregelung (*Clock disciplin algorithm*) weitergegeben



- Zeit in verteilten Systemen ist eine zentrale Herausforderung
- Wenn Aussagen zur kausalen Ordnung von Ereignissen benötigt werden, bietet sich der Einsatz von logischen Uhren an
 - Logische Uhren nach Lamport, erweiterbar zu einer totalen Ordnung auf alle Ereignisse, die kausale Beziehungen respektiert
 - Vektoruhren zur exakten Erfassung von kausalen Beziehungen
- Das Network Time Protocol (NTP) bietet die Möglichkeit, lokale Uhren mit für viele Zwecke ausreichender Genauigkeit an die offizielle Zeit zu synchronisieren
 - Hierarchisches Synchronisationsnetz
 - Selbstorganisierend und fehlertolerant

