Verteilte Betriebssysteme

6. Kapitel
Uhrensynchronisation

Matthias Werner
Professur Betriebssysteme

6.1 Motivation

- Im letzten Kapitel haben wir uns Gedanken über verschiedene Semantiken für eine **korrekte** Reihenfolge gemacht
- Im alltäglichen Leben definiert man Reihenfolgen i.d.R. über **Zeit**
- Es wird eine **globale Zeit** angenommen, die von **Uhren** abgelesen wird

Globale Zeit

- Die Bestimmung der Zeit und die Messung von Zeitdauern ist unverzichtbar zur Koordination menschlicher Aktivitäten
- Hierfür müssen unsere Uhren synchronisiert sein
  - Synchronisation der Uhren mittels Kirchturmuhr, Telegraphie, Radio, GPS
  - Uhrensynchronisation ermöglichte in der Schifffahrt erst die Längengradbestimmung
- **Die Existenz einer globalen Zeit haben wir verinnerlicht**

Bildquelle: NMM London, MoD Art Collection

Diskussion

Das Konzept der Existenz einer globale Zeit ist nur eine **Modellvorstellung** aus der Newtonschen Physik. In anderen Ansätzen, wie der Minkowski-Raum-Zeit[1] ist eine globale Zeit nicht darstellbar.

[1] Modell(!) aus der Relativitätstheorie

Zeit in Rechnern

- Nutzung von Zeitstempeln, um ...
  - ... die Aktualität von Daten zu bewerten
  - ... Leistungsmessungen durchzuführen
  - ... die Gültigkeit von Zugriffsberechtigungen zu bestimmen
  - ... eine Totalordnung von Ereignissen zu bewirken (z.B. für Synchronisation, Debugging und Audit)
  - ... Sensordaten auszuwerten und Aktuatoren zu steuern

6.2 Zeit und Uhren

Begriffe

- Eine **Uhr** bildet die Realzeit t auf einen Zeitstempel $C(t)$ ab
- **Auflösung**
  Kleinste Zeitspanne um die sich zwei Werte einer Uhr unterscheiden können
  ➡ **Tickdauer**
- **Offset**
  Abweichung der Uhr von der Realzeit zu einem Zeitpunkt, d.h. $C(t) - t$
- **Drift**
  Abweichung der **Geschwindigkeit** der Uhr von der Realzeit
  - $\pm 10^{-6}$ bei Quarzuhren bzgl. UTC (2s pro Monat)
  - $\pm 10^{-18}$ bei Atomuhren bzgl. der idealen SI-Sekunde

Lokale Uhren

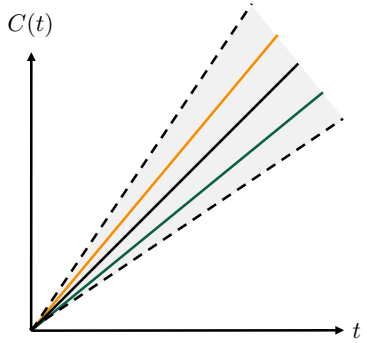
- Jeder lokale Knoten hat eine eigene ungenaue, digitale Uhr
- Die Uhren-Drift ist unterschiedlich
- Ohne Synchronisierung können sich die Werte der Uhren beliebig weit voneinander entfernen ➡ **Uhrensynchronisation** notwendig
- Nur durch **Nachrichtenaustausch** möglich
- Hierbei spielt die **Nachrichtenlaufzeit** eine wichtige Rolle
- **Interne Synchronisation**: Teilnehmer im verteilten System einigen sich, kein Bezug zu physischer Zeit
- **Externe Synchronisation**: Zeitquelle außerhalb des eigenen Systems, eigene Zeitzone berücksichtigen

Korrekte Uhren

- Eine **korrekte Uhr** hat eine beschränkte Drift ρ_{max}

$$\frac{1}{1+\rho_{max}} \leq \frac{dC(t)}{dt} \leq 1+\rho_{max} \quad (1)$$

- **Anmerkung**: Drift wird nur im Bereich Δt betrachtet, der größer als die Zeitauflösung ist, bzw. eine kontinuierliche Uhr angenommen



zu schnell, $\frac{dC}{dt} > 1$

exakt, $\frac{dC}{dt} = 1$

(Wert der Uhr steigt linear mit der realen Zeit)

zu langsam, $\frac{dC}{dt} < 1$

- Uhren, die **stets** korrekte Zeit anzeigen, sind **perfekte** Uhren (Modellvorstellung!)

Diskussion

- Auch eine zu schnelle oder zu langsame Uhr wird als **korrekt** angesehen
- Bedingung wird häufig angegeben als:

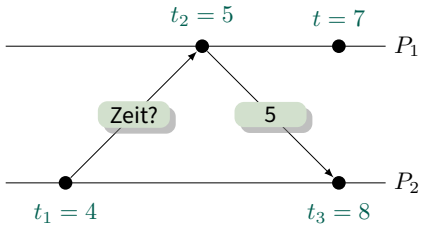
$$1 - \rho \leq \frac{dC(t)}{dt} \leq 1 + \rho \quad (2)$$

- Dies ist aber nur bei sehr kleiner Drift sinnvoll ($\rho \ll 1$)
  - Betrachten $\rho = 1$
  - **Bedingung** (1): Uhr ist korrekt, solange sie nicht schneller als doppelt so schnell wie die perfekte Uhr und nicht langsamer als halb so schnell wie sie läuft
  - **Bedingung** (2): Uhr ist korrekt, solange sie nicht schneller als doppelt so schnell wie die perfekte Uhr läuft, aber sie darf **stehen bleiben**
    - Bei $\rho > 1$ dürfte eine korrekte Uhr sogar **rückwärts laufen**
  - Für meisten praktischen Belange ist (2) aber ausreichend

6.3 Interne Synchronisation

6.3.1 Uhrensynchronisation nach Cristian

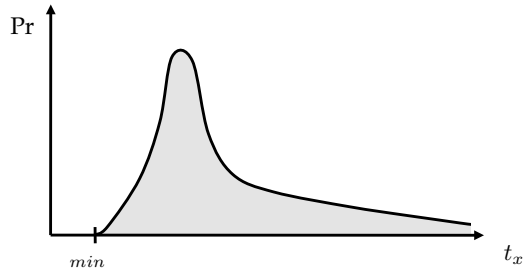
- **Szenario**: Prozess P_2 will seine Uhr auf P_1 abstimmen
- Vorläufige Annahme: Konstante Nachrichtenlaufzeit t_x
- P_2 passt seine Uhr um $t_2 + t_x - t_3$ an
- Soll die maximale Abweichung kleiner als Δ bleiben, so ist eine erneute Synchronisation spätestens nach $\frac{\Delta}{\rho}$ notwendig



➡ P_2 stellt Uhr um 1 ZE zurück

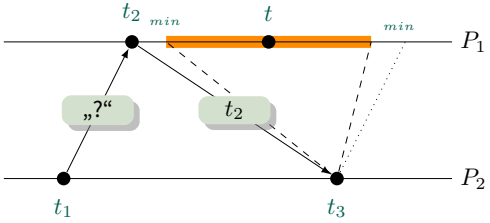
Unvorhersehbare Nachrichtenlaufzeiten

- In der Realität sind Nachrichtenlaufzeiten meist lastabhängig und unbeschränkt
  - Die Laufzeit ist bei höherer Last größer als bei niedriger Last
  - Die Laufzeit kann beliebig lang sein
  - Minimale Laufzeit min kann nicht unterschritten werden
- In diesem Fall führt das bisherige Verfahren zu einem **Fehler** bei der Anpassung



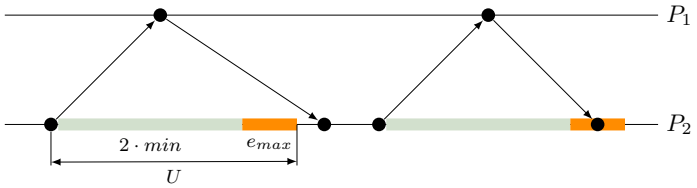
Unvorhersehbare Nachrichtenlaufzeiten (Forts.)

- **Round Trip Time** (RTT) $D \stackrel{def}{=} (t_3 - t_1)$
- Sei t die lokale Zeit von P_1, wenn P_2 die lokale Zeit t_3 hat
- t liegt im Intervall $[t_2 + min, t_2 + D - min]$
- Die „bestmögliche“ Schätzung für t ist die Intervallmitte $t_2 + \frac{D}{2}$
- Daher korrigiert P_2 seine Uhr um $t_2 + \frac{D}{2} - t_3$
- Der maximaler Fehler dieser Anpassung ist $e_{max} = \frac{D}{2} - min$



Probabilistische Begrenzung des max. Fehlers

- **Idee:**
  1. Akzeptiere Wert der Referenzuhr nur, wenn die RTT D einen bestimmten Grenzwert U nicht überschreitet ➡ $D \leq U$
  2. Wiederhole bei Fehlschlag den Versuch maximal k-mal, jeweils nach einer Wartezeit $t_W > U$
- Sei $\Pr_U$ die Wahrscheinlichkeit, dass $D > U$ bei beliebigen Versuch (Fehlschlag)
- Wahrscheinlichkeit für n Fehlversuche: $(\Pr_U)^n$
- Erwartungswert für Anzahl von Versuchen: $E = \frac{1}{1-(\Pr_U)}$



Maximales Synchronisationsintervall

- Zwei korrekte Uhren mit der Drift ρ sollen nicht mehr als Δ auseinanderlaufen
  ➡ Nach welcher Zeit muss spätestens eine Synchronisation stattgefunden haben?
- **Annahme:** zu $t = 0$ sind die Uhren synchron
- **Betrachten Worst Case:** Maximale Abweichung der Uhren ➡ Uhren C_1 und C_2 gehen maximal vor/nach

$$\frac{dC_1(t)}{dt} = 1 + \rho \qquad \frac{dC_2(t)}{dt} = \frac{1}{1+\rho}$$

Worst case zum Zeitpunkt t: $\quad C_1(t) = (1+\rho)t \qquad C_2(t) = \frac{t}{1+\rho}$

Maximales Synchronisationsintervall (Forts.)

Forderung: $C_1(t) - C_2(t) \overset{!}{\leq} \Delta$

Folglich:
$$(1+\rho)t - \frac{t}{1+\rho} \leq \Delta$$
$$\frac{(1+\rho)^2 t - t}{1+\rho} \leq \Delta$$
$$t\frac{2\rho+\rho^2}{1+\rho} \leq \Delta$$

Synchronisationszeit

$$t \leq \Delta\frac{1+\rho}{2\rho+\rho^2}$$

- Uhren müssen also spätestens nach $\Delta\frac{1+\rho}{2\rho+\rho^2}$ synchronisiert worden sein
- Für sehr kleine ρ: $t \approx \frac{1}{2\rho}\Delta$

Maximales Synchronisationsintervall (Forts.)

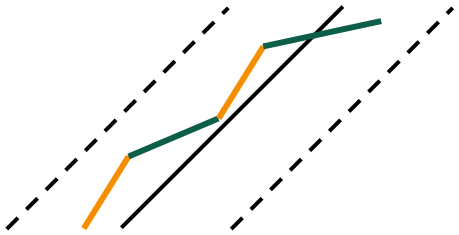
- Wir geben die Annahme auf, dass die Uhren zu $t = 0$ synchron sind ➡ probabilistische Synchronisation, siehe Folie 6-12
  - Nach der letzten erfolgreichen Synchronisation ist die Abweichung maximal $e_{max} = \frac{D}{2} - min$
  - Es muss so synchronisiert werden, dass nicht mehr als $\Delta' = \Delta - e_{max}$ bis zum erfolgreichen Versuch verstreicht
  - Da die Synchronisation selbst Zeit benötigt, muss sie rechtzeitig gestartet werden

Zeit bis zum Starten der nächsten Synchronisation (nach letzter Synchronisation)

$$t \approx \frac{1}{2\rho}(\Delta - e_{max}) - \frac{t_W}{1 - \mathrm{Pr}_U} \tag{3}$$

Anpassung der lokalen Uhrzeit

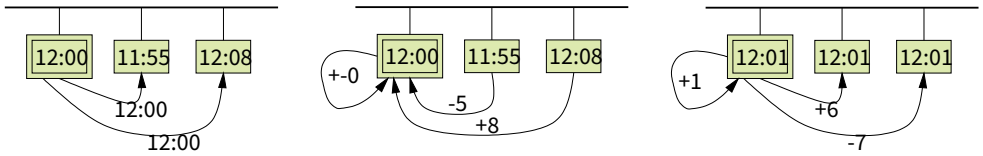
- **Anforderungen** (aus Korrektheitsforderungen)
  - Große Sprünge der Uhrzeit sollen vermieden werden
  - Die Uhr darf nie rückwärts laufen
- **Lösung**
  - Die lokale Uhr wird solange etwas langsamer bzw. schneller betrieben, bis der Offset ausgeglichen wurde



Berkeley-Algorithmus

- Algorithmus zur **internen** Uhrensynchronisation
  - Synchronisiert eine Menge von Uhren untereinander
  - Nicht notwendigerweise Bezug zur Realzeit vorhanden
  - Enwickelt 1989 an der University of California, Berkeley [GuZa87]
- **Funktionsweise**
  - Das System besteht aus mehreren Uhren
    - Keine Annahme, dass eine davon die „wahre“ Zeit hat
  - Wenn nötig, Wahl eines Anführers ➡ siehe Kapitel 9
  - Der Anführer ermittelt für jede andere Uhr deren Abweichung ➡ Berechnung mit Hilfe der RTT analog zum Cristian-Algorithmus
  - Korrektur um die Differenz vom Durchschnittswert und Abweichung ➡ keine Übermittlung von absoluter Zeit
    - Kein zusätzlicher Fehler durch RTT

Berkeley-Algorithmus (Forts.)



$$Avg = \frac{0 - 5 + 8}{3} = 1$$
$$L = 1 - 0 = 1$$
$$F_1 = 1 - (-5) = 6$$
$$F_2 = 1 - 8 = -7$$

- Um den Einfluss fehlerhafter Uhren zu reduzieren, werden Uhren mit zu großer Zeitdifferenz ignoriert
- Jedoch erhalten auch solche Uhren einen Korrekturwert

Berkeley-Algorithmus (Forts.)

- Auswahl eines Schwellwertes für die Abweichung
  - Aussortieren stark abweichender Maschinen (Durchschnittsberechnung)
  - Zu klein: Nur Synchronisation einer kleinen Teilmenge der Slaves
  - Zu groß: Fehlerhafte Slaves verschlechtern die Präzision der synchronisierten Zeit
  - Wert bei $20ms$ in der Originalpublikation von 1987
- Auch eine interne Synchronisation soll sich typischerweise an der ‘echten’ Uhr, also der globalen Zeit orientieren
- Was ist das überhaupt?

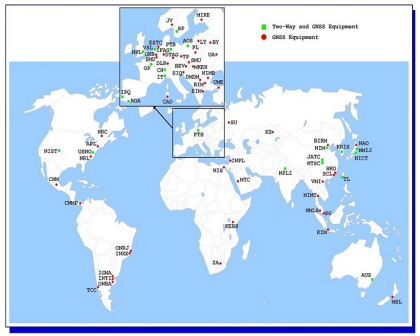
6.4 Externe Synchronisation

Globale Zeit

- Älteste Grundlage der Zeitmessung: Astronomie
- *UT - Universal Time*: Astronomische mittlere Sonnenzeit
  - Ermittelt durch astronomische Beobachtung
  - Eine mittlere Solarsekunde ist 1/86400 eines mittleren Solartages
  - Folgt den Schwankungen der Erdrotation ➡ Variierende Zeiteinheit
  - Solartag wird jedes Jahrhundert ca. 1.7ms länger (Mondeffekte), vor 300 Millionen Jahren hatte ein Jahr noch 400 Tage.

Globale Zeit (Forts.)

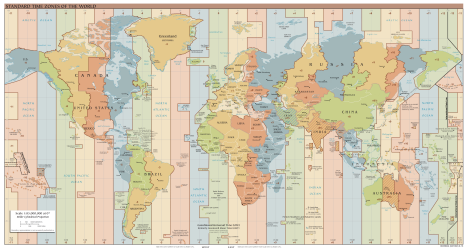
- *TAI - Temps Atomique International*: Internationale Atomzeit
- Basiert auf Strahlungsfrequenzen des Cäsium 133 Atoms
- Gewichtetes Mittel von ca. 260 Atomuhren weltweit
- Standardabweichungen im Bereich von wenigen Nanosekunden
- Zählung seit 1.1.1958, zunehmende Abweichung zwischen UT und TAI
- Effekte durch Atmosphäre, Strahlung schwarzer Löcher und Gravitationseffekte
- Synchronisation durch Messung von GPS-Signalen und Satellitenkommunikation



Quelle: http://www.bipm.org/

Globale Zeit (Forts.)

- *UTC*: Koordinierte Weltzeit
  - *Temps universel coordonné* bzw. *Coordinated Universal Time*
  - Länge einer UTC-Sekunde konstant, definiert aus internationaler Atomzeit
  - Schaltsekunden zum Ausgleich der UT-Schwankungen ($\Delta > 0.6s$)
  - Verbreitung per Radiosignal ($\pm 10ms$) oder Satellit ($\pm 0.5ms$)
  - Zeitzonen relativ zu UTC

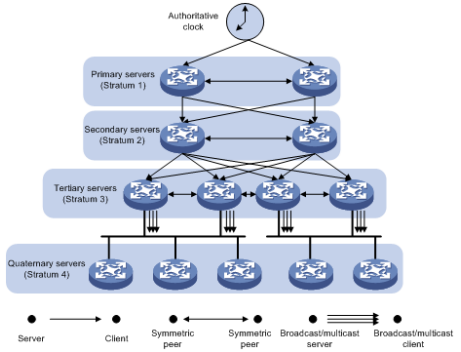


Bildquelle: Wikipedia

Network Time Protocol – NTP

- Protokoll zur Synchronisation von (Rechner-)uhren mittels Internet
- Eines der ältesten UDP-Protokolle (RFC 958 aus 1985), mittlerweile Version 4 (RFC 5905)
  - Mechanismus zur **externen** Uhrensynchronisation
  - Große und variierende Verzögerungen von Nachrichten
  - Unterschiedliche Qualität der Zeit-Server
  - Redundante Kommunikationspfade
  - Skalierbarkeit für große Anzahl periodisch anfragender Clients
  - Schutz durch Authentifizierung
- Baum von Servern bis zu den Endnutzergeräten (`ntpq -pn`)
- Ausgangspunkt ist UTC-Zeitquelle
- Für 99% aller NTP-Clients beträgt der Fehler zur Referenzzeit $\Delta < 30$ ms

Network Time Protocol – NTP (Forts.)



Bildquelle: http://www.h3c.com.hk

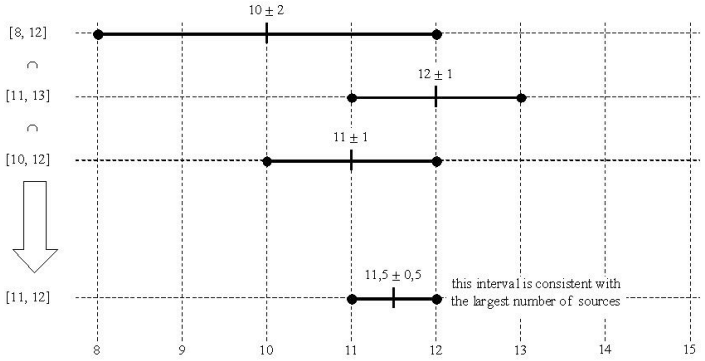
David Mills

„Boiled down, really the only thing the protocol does is occasionally read the clock in another machine, tell whether the system clock is fast or slow, and nudge the clock oscillator one way or the other, just like the Big Ben timekeepers in London. Of course, the devil is in the details“

- Synchronisation auf einer Ebene (Stratum): ähnlich Berkeley-Algorithmus
- Synchronisation between Ebenen ähnlich Cristian

NTP-Detail: Marzullo-Algorithmus

- Teilalgorithmus zur Bestimmung eines Zeitintervalls bei überlappenden Quellen
- **Annahme**: Zeitquellen bieten Zeit mit Vertrauensintervallen
- **Ziel**: Kleinstes Intervall mit der größten Anzahl von übereinstimmenden Quellen



NTP-Detail: Marzullo-Algorithmus (Forts.)

- Zeitdaten als Intervall $[c - r, c + r]$
- Zwei Tupel der Form *(offset, type)* pro Quelle: $(c - r, -1)$ und $(c + r, 1)$
- Liste der Tupel nach ersten Wert sortieren
- Anschließend:

```
INIT: best=0 /* largest number of overlapping intervals */
INIT: count=0 /* current number of overlapping intervals */
DO LOOP FOR ALL Tupels i in List
  count := count - type[i]
  IF count > best THEN
    best := count
    best_start := offset[i]
    best_end := offset[i+1]
  END IF
END LOOP
RETURN [best_start, best_end]
```


6.5 Grenzen der Synchronisation

- Wie gut können Uhren synchronisiert werden? ➡ [LuLy84]
- **Annahmen**:
  - Lokale Uhren haben keine Drift
  - Kommunikationszeiten im Intervall $[t, t + \varepsilon]$
  - Stand der Uhren anfangs unbekannt

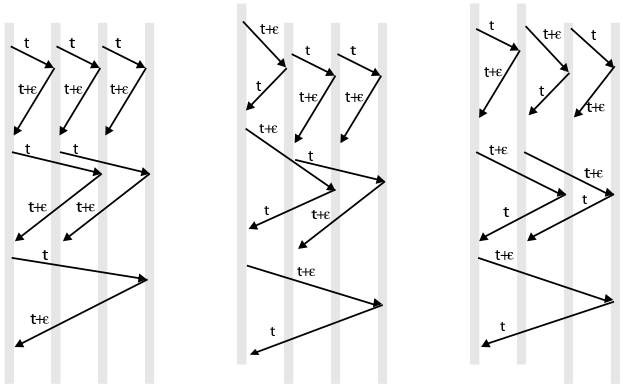
Theorem 1

Es gibt keinen Algorithmus, der n Uhren enger als auf ein Intervall der Größe $\varepsilon(1 - \frac{1}{n})$ garantiert synchronisiert.

- Grenze ist hart: Es gibt Algorithmen, die Grenze garantieren.

Grenzen der Synchronisation (Forts.)

- Betrachten Abläufe im Worst Case
- Alle Abläufe e^i, $i = 1, \ldots, n$ sind lokal **nicht unterscheidbar**



Grenzen der Synchronisation (Forts.)

- Sei C_i der lokale Uhrenwert von Knoten i und δ die Abweichung untereinander

$$C_n \leq C_1 + \delta \quad \text{für Ablauf } e^1 \tag{4}$$
$$C_{i-1} + \varepsilon \leq C_i + \delta$$
$$\text{bzw. } C_{i-1} \leq C_i + \delta - \varepsilon \quad \text{für Ablauf } e^i, i = 2, \ldots, n \tag{5}$$

Addiere (4) und (5)

$$\sum_{i=1}^{n} C_i \leq \sum_{i=1}^{n} C_i + n \cdot \delta - (n-1) \cdot \varepsilon$$
$$(n-1) \cdot \varepsilon \leq n \cdot \delta$$
$$\delta \geq \varepsilon \left(1 - \frac{1}{n}\right) \qquad \square$$

Inkorrekte Uhren

- **Inkorrekte Uhren** können jede beliebige Zeit anzeigen – und das bei jeder Abfrage!
  - Auch Sprünge oder rückwärts laufende Uhren möglich
  - Jede Mittelwertbildung schlägt fehl
  - **Lösung**: **Byzantinisches Agreement**
    ➡ Vorlesung „Verlässliche Systeme“
  - **Anforderung**: weniger als ein Drittel aller Uhren dürfen fehlerhaft sein
- **Anwendungsbeispiel**: Synchronisation in FlexRay
  (Kommunikationsplattform für Kraftfahrzeuge)

Literatur

[Cri89] Cristian , F. : „Probabilistic Clock Synchronization“. *Distributed Computing*, 3(3)1989, 146–158

[GuZa87] Gusella , R. und S. Zatti: *The Accuracy of the Clock Synchronization Achieved by TEMPO in Berkeley UNIX 4.3BSD*, technical report, EECS Department, University of California, Berkeley

[LuLy84] Lundelius , J. und N. A. Lynch: „An Upper and Lower Bound for Clock Synchronization“. *Information and Control*, 62(2/3)1984, 190–204