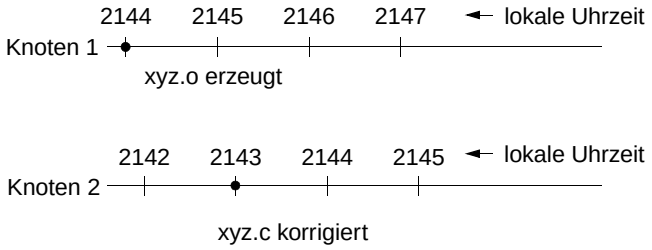
Uhrensynchronisation

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

Zeit als Mittel der Reihenfolgebestimmung

Beispiel: **make**

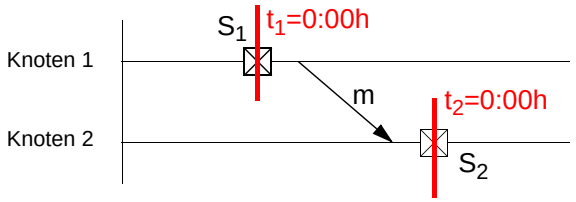


Modifikation von xyz.c wird nicht erkannt, Datei wird nicht neu übersetzt!

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: Inkonsistenz möglich, falls die lokalen Uhren voneinander abweichen:



Auswirkung von m in S_2 enthalten, aber nicht in S_1!

Zeit als Mittel der Reihenfolgebestimmung

Beispiel: Monitoring und Debugging

- Verteilte Datenerfassung zu Protokoll- oder Debuggingzwecken
- Tatsächliche Reihenfolge kann nicht wiederhergestellt werden, wenn lokal generierte Zeitstempel nicht synchron sind

Probleme der Uhrensynchronisation

- Es gibt keine völlig identischen physikalische Uhren
  - Abweichende Initialisierung (konstantes Offset)
  - Abweichende Ganggeschwindigkeit (Frequenzfehler)
  - Unterliegt Umgebungseinflüssen (z.B. Bauteilalterung, **Temperaturabhängigkeit**)

  $\Rightarrow$ Ohne Synchronisierung kann Fehler immer größer werden!
- Gemeinsame Uhr für alle Knoten eines verteilten Systems (meist) nicht realisierbar
- „Zentrale Referenzuhren“ (Funk, z.B. DCF77 Langwellensender in Mainflingen, GPS, WWV) 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

  $\Rightarrow$ Synchronisation bei Kommunikation
- Anmerkung: Die im folgenden beschriebene Verfahren setzen voraus, dass die Kommunikation nur durch Nachrichtenaustausch über ein Kommunikationsnetz erfolgt!

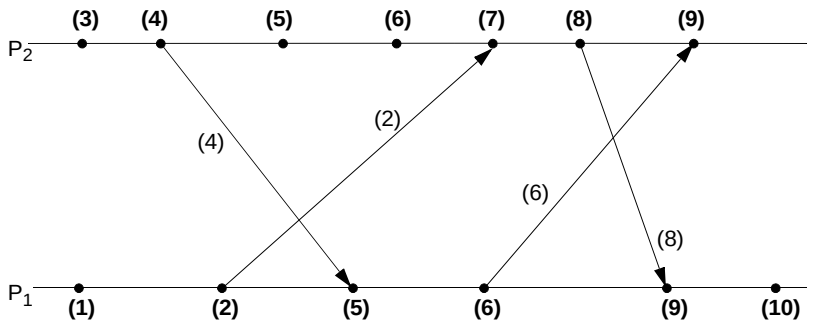
Logische Uhren

Beschreibung des Algorithmus

Jeder Prozess i verfügt über eine Uhr C_i , die 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

Logische Uhren



Logische Uhren

Eigenschaften

- Potentielle kausale Abhängigkeit eines Ereignisse E_2 vom Ereignis E_1 (Kurznotation: $E_1 \rightarrow E_2$): $E_1 \rightarrow E_2 \Rightarrow t(E_1) < t(E_2)$
- Die Umkehrung: $t(E_1) < t(E_2) \Rightarrow E_1 \rightarrow E2$ 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 Prozessor erhält eindeutigen Identifikation
- Zeitstempel bestehend aus Prozessorn. i und lokaler Zeit C_i.
- Anordnung:

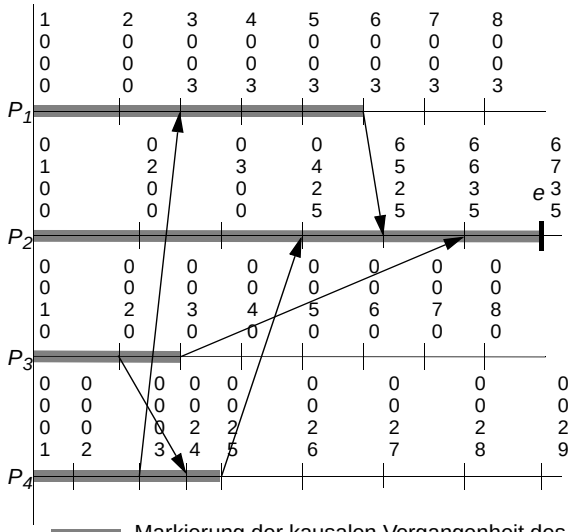
$$(C_i, i) < (C_k, k) \Leftrightarrow C_i < C_k \text{ oder } (C_i = C_k \text{ und } i < k)$$

Logische Uhren — Vektoruhren

- Jeder Knoten besitzt nun eine lokale Uhr, die aus einem Vektor aus 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:

$$C_i[i] + +$$
$$\text{für } k = 1 \ldots N: \quad C_i[k] := max(C_i[k]; t[k])$$

Logische Uhren



Logische Uhren

Eigenschaften

- Erzeugt eine kausale Ordnung: $t(E_1) < t(E_2) \Leftrightarrow E_1 \rightarrow E_2$

  Definition von $<$ mit $t(E_1) := (a_1, ..., a_n)$ und $t(E_2) := (b_1, ..., b_n)$:

$$t(E_1) < t(E_2) \Leftrightarrow (\forall i : a_i \leq b_i) \wedge (\exists i : a_i < b_i)$$

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!)

Logische Uhren

1. Verbesserung (Singhal-Kshemkalyani)

- Übertragung nur der Werte, die sich seit letzter Nachricht an diesen Prozess geändert haben
- Effektiv bei vielen Prozessen oder örtlicher Lokalität der Kommunikation

Logische Uhren

2. Verbesserung (Fowler-Zwaenepoel)

- Knoten unterhalten keine **aktuellen** Vektoruhren; statt dessen nur direkt abhängige Ereignisse im Abhängigkeitsvektor registriert. D.h. es wird jeweils nur der eigene Eintrag im Abhängigkeitsvektor gesendet
- Ermittlung der echten Vektor-Zeitstempel nachträglich durch Rückverfolgung der direkt abhängigen Ereignisse
- Geeignet nur, wenn kein aktueller Zeitstempel notwendig ist (Debugging, Logging)

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 Sonne festgelegt)
- Genauigkeit von Cs-Uhren: bis ca. 10^{-14} (entspricht ca. 0.8 ns/Tag!)
- **UT1**: über astronomische Beobachtungen definierte Zeit (mittlere Sonnenzeit am Nullten Längengrad)
- **UTC** (Universal Time Coordinated): Skala basierend auf Atomsekunde; $|UTC - UT1| < 0.9s$ über Schaltsekunden (bisher 24, zuletzt 2008)

Synchronisation von physikalischen Uhren

Verbreitung der amtlichen Zeit: Normalfrequenz/ Zeitzeichensender

- Langwelle: z.B. DCF77 (Mainflingen, D); WWVB (Boulder, US); maximale Genauigkeit bis ca. $50\mu s$
- Kurzwelle: z.B. WWV (Colorado), WWVH (Hawaii); maximale Genauigkeit bis ca. $1ms$
- GPS-Satelliten: maximale Genauigkeit ca. $0.3\mu s$

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

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

Aufgabe des Algorithmus

- Vermeidung einer Akkumulation einer immer größer werdenden Abweichung zwischen mehreren Uhren

Konvergenz-Algorithmus CNV

Ablauf

- Jeder Prozess liest die Uhr jedes anderen
- 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

- 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 Prozessor kann die Uhr eines anderen korrekten Prozessor 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 $\Rightarrow |C_{p,r} - Cq, r| < 3\delta$
- Alle Prozesse p stellen ihre Uhr auf $\sum_i (C_{p,i})/n$

Konvergenz-Algorithmus CNV

Plausibilitätsbetrachtung

- Schlechtester Fall:

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

- 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
  - Fehler durch nicht gleichzeitiges Ablesen der Uhren

Das Network Time Protocol - NTP

- Ausführliche Informationen zu NTP im WWW:
  `http://www.ntp.org` (Offizielle NTP-Homepage)
  `http://www.eecis.udel.edu/~mills` (Homepage David Mills)
- Geschichte
  - Entwickelt seit 1982 (NTP v1, RFC 1059) unter Leitung von David Mills
  - Seit 1990 NTP v3 (teilweise immer noch in Verwendung)
  - Aktuelle Version NTP v4, seit 1994

Das Network Time Protocol - NTP

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. $0.01s$ in WANs, $< 1ms$ in LANs
- NTP-Dämon auf fast allen Rechnerplattformen verfügbar, von PCs bis Crays; Unix, Windows, OS X, eingebettete Systeme
- Fehlertolerant

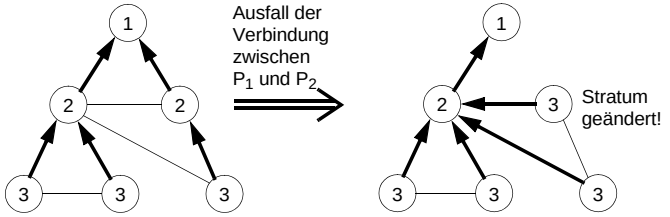
Das Network Time Protocol - NTP

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.

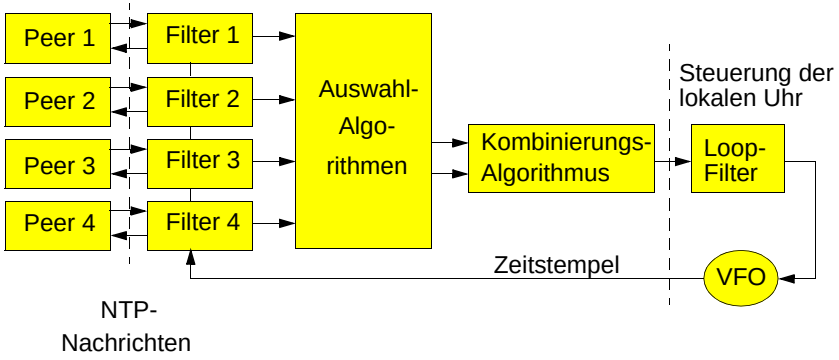
Das Network Time Protocol - NTP

- Stratum:
  - 1: primärer Zeitgeber
  - $i > 1$: synchronisiert mit Zeitgeber des Stratums $i - 1$
- Stratum kann dynamisch wechseln:



Das Network Time Protocol - NTP

Architektur-Überblick

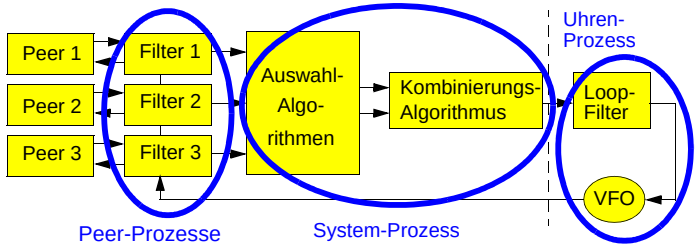


Das Network Time Protocol - NTP

- **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 richtiggehende Uhren („truechimers") zu erkennen und falsche Uhren („falsetickers") auszufiltern, 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

Das Network Time Protocol - NTP

NTP-Prozesse

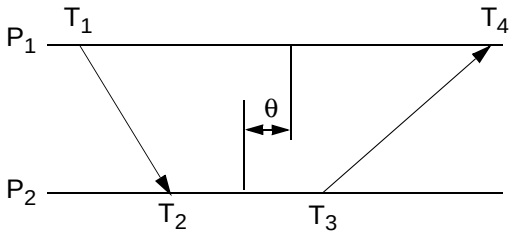


- **Peer-Prozesse**: Unabhängig in periodischen Abständen; Periode dynamisch durch System-Prozess und durch entfernte Server bestimmt
- **System-Prozess**: Periodisch, Abstände dynamisch aus Phasen-Jitter und Stabilität der lokalen Uhr bestimmt
- **Uhren-Prozess**: Periodisch in 1s-Intervallen (Regelung VFO-Phase und -Frequenz)

Das Network Time Protocol - NTP

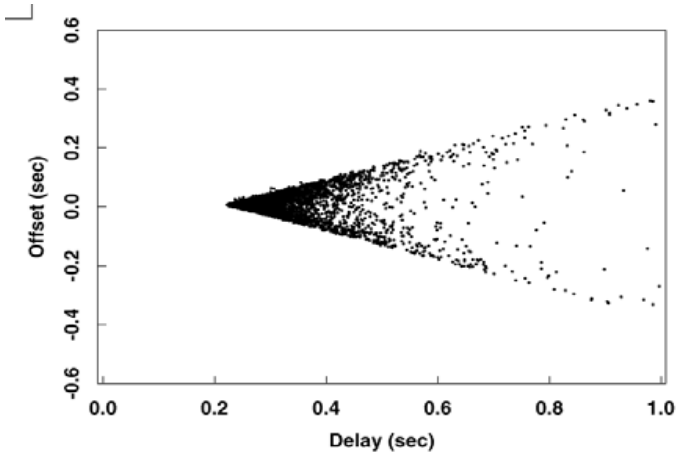
Offset-Messung

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



- Reine Nachrichtenlaufzeit: $\delta = (T4 - T1) - (T3 - T2)$
- 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: $\delta/2$

Das Network Time Protocol - NTP



Bei kleinstem Delay δ ist der gemessener Offsetwert θ am genauesten!

Das Network Time Protocol - NTP

Aufbau der NTP Datenpakete

LI	VN	Mode	Strat	Poll	Prec
Root Delay					
Root Dispersion					
Reference Identifier					
Reference Timestamp (64)					
Originate Timestamp (64)					
Receive Timestamp (64)					
Transmit Timestamp (64)					
Extension Field 1 (optional)					
Extension Field 2 (optional)					
Key/Algorithm Identifier					
Message Hash (64 or 128)					

LI	leap indicator
VN	version number
Strat	stratum (0..15)
Poll	poll interval (log2)
Prec	precision (log2)

NTP Zeitstempel-Format

Seconds (32)	Fraction (32)

Sekunden seit dem 1.1.1900

Extension Field: nur NTPv4

Das Network Time Protocol - NTP

Der Peer-Filter-Algorithmus: Ausgangsdaten

- Bei jeder Messung ermittelte Werte:
  - Offset θ, Laufzeit δ
  - Abschätzung des Messfehlers:
    $\epsilon = \mathit{Lesegenauigkeit} + \mathit{MAXDRIFT} * (T4 - T1)$
- 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)

$i = 7..1 : (\delta_i, \theta_i, \epsilon_i) = (\delta_{i-1}, \theta_{i-1}, \epsilon_{i-1} + \mathit{MAXDRIFT} * \mathit{verstrichene\ Zeit})$
$i = 0 : (\delta_0, \theta_0, \epsilon_0) = \mathit{neuer\ Messwert}$

Das Network Time Protocol - NTP

Der Peer-Filter-Algorithmus: Filterung

- **Eingangsdaten**: Liste von Messwerten $(\delta_i, \theta_i, \epsilon_i)$
- **Sortierung** der Messwerte nach $\epsilon_j + \delta_j/2$;
  Minimaler Wert wird als Referenz verwendet
- **Berechnung** der Filter-Streuung ϵ_σ als gewichtete Abweichung des Messwerts relativ zu den anderen, sortierten Messwerten (als Maß für die Genauigkeit der Messwerte)

$$\epsilon_\sigma = \sum_{i=0}^{I} |\theta_i - \theta_0| v^i \ \mathit{mit}\ \text{v=0.5}\ (\mathit{experimentell})$$

- Das **Ergebnis** ist für jedes Peer ein Tripel aus Delay, Offset, Dispersion:

$$(\delta, \theta, \epsilon) = (\delta_0, \theta_0, \epsilon_0 + \epsilon_\sigma)$$

NTP v4: Zusätzlich Berechnung der Varianz der Messwerte

Das Network Time Protocol - NTP

Variablen-Berechnung für Systemprozess

- Für jedes Peer werden folgende Variablen berechnet:

$\Theta = \theta$

$\Delta = \mathit{RootDelay} + \delta$

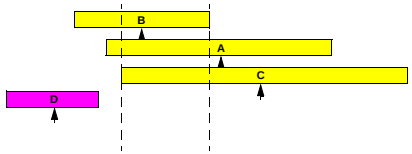
$E = \mathit{RootDispersion} + \epsilon + (\mathit{Alter\ der\ Messung}) * \mathit{MAXDRIFT}$

$\Rightarrow$ Tripel $(\Theta_i, \Delta_i, E_i)$ für das Peer i

- Weitergabe an den Systemprozess, dieser berechnet daraus:
  - **Synchronisierungsdistanz**: $\Lambda_i = E_i + \Delta_i/2$
  - **Korrektheitsintervall**: $[\Theta_i - \Lambda_i, \Theta_i + \Lambda_i]$

Das Network Time Protocol - NTP

Auswahl-Algorithmus



- Trennung von „truechimers" und „falsetickers"
- DTS (Digital Time Service, Vorgänger-Algorithmus, einfacher): Ermittle Durchschnitt mit den meisten überlappenden Korrektheitsintervallen. 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

Das Network Time Protocol - NTP

Clustering-Algorithmus

- **Ziel**: Weitere Selektion von möglichst „guten" Referenzen (Stratum und Synchronisierungsdistanz möglichst klein)
- Sortierung der Liste der Referenzen nach der Metrik
  *Stratum * MAXDISPERSE + Synchronisierungsdistanz*
- Für jedes Peer k wird eine gewichtete Streuung $\epsilon_{\xi k}$ relativ zu den anderen Peers berechnet
  $$\epsilon_{\xi k} = \sum_{i=0}^{n} |\theta_i - \theta_k| w^i \text{ mit } w = 0.75$$
- Der Knoten mit maximalen $\epsilon_{\xi k}$ wird entfernt, und das ganze von vorne wiederholt, solange
  - es in der Liste mehr als *MINCLOCK* Knoten gibt
  - das maximale $\epsilon_{\xi k}$ größer als das minimale E_i ist

Das Network Time Protocol - NTP

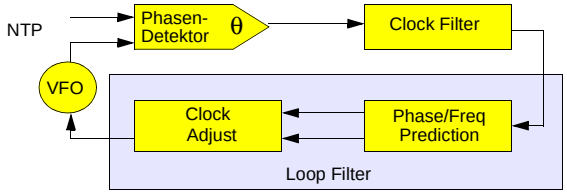
Kombinierungsalgorithmus

- Die übrigbleibenden 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

Das Network Time Protocol - NTP

Uhrenregelung

- Regelungsprinzip: Hybride PLL (phase locked loop)/ FLL (frequency-locked loop)-Regelschleife
- Regelungstechnische Modellierung:



- Ziel der hybriden Regelung:
  - Frequenzregelung zum Ausgleich von Frequenzfehlern und langfristigen Schwankungen
  - Phasenregelung zur Korrektur von kurzfristigen Schwankungen

Das Network Time Protokol - NTP

Praktische Beispiele

- Notebook ...

```
ntpq> peers
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
*media1r.euro.ap 17.72.133.54     2 u  491  256  376  136.542   71.883   7.880

ntpq> rv
assID=0 status=06b4 leap_none, sync_ntp, 11 events, event_peer/strat_chg,
version="ntpd␣4.2.2@1.1532-o␣Mon␣Sep␣24␣01:42:27␣UTC␣2007␣(1)",
processor="i386", system="Darwin/9.3.0", leap=00, stratum=3,
precision=-20, rootdelay=140.448, rootdispersion=161.705, peer=3389,
refid=17.72.133.45,
reftime=cbfff484.c127354c  Sun, Jun 15 2008 22:05:24.754, poll=8,
clock=cbfff671.d8ddc88f  Sun, Jun 15 2008 22:13:37.847, state=4,
offset=71.883, frequency=-2.957, jitter=7.880, noise=14.367,
stability=0.107, tai=0
```

Das Network Time Protokol - NTP

Praktische Beispiele

- Rechner im CIP-Pool

```
faui02d: 21:28 ~ [46] > ntpq
ntpq> peers
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
 ntp0-rz.rrze.un .STEP.          16 u    - 1024    0    0.000    0.000   0.000
+ntp1-rz.rrze.un .DCFp.           1 u  913 1024  377    1.631    0.649   0.248
*ntp2-rz.rrze.un .GPS.            1 u  199 1024  377    2.654    0.081   1.443
+ntp3-rz.rrze.un .DCFp.           1 u   25 1024  377    1.623    0.762   0.025
 LOCAL(0)        .LOCL.           7 l   52   64  377    0.000    0.000   0.001


ntpq> rv
assID=0 status=06f4 leap_none, sync_ntp, 15 events, event_peer/strat_chg,
version="ntpd␣4.2.2p4@1.1585-o␣Sun␣Mar␣␣4␣13:21:35␣UTC␣2007␣(1)",
processor="x86_64", system="Linux/2.6.25.4", leap=00, stratum=2,
precision=-20, rootdelay=2.654, rootdispersion=35.644, peer=1788,
refid=131.188.3.222,
reftime=cbffe722.47039c31  Sun, Jun 15 2008 21:08:18.277, poll=10,
clock=cbffec16.b86bdaa6  Sun, Jun 15 2008 21:29:26.720, state=4,
offset=0.434, frequency=-11.023, jitter=1.494, noise=0.394,
stability=0.022, tai=0
```


Zusammenfassung

- Zeit in verteilten Systemen ist ein 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