RUS
Rechenzentrum
Universität Stuttgart
Prof. Dr.-Ing. R. Rühle
Allmandring 30, 70550 Stuttgart

ICA II
Institut für Computeranwendungen
Abt. Computersimulation
und Visualisierung

AIM
Anwendungen der Informatik
im Maschinenwesen

FORSCHUNGS- UND ENTWICKLUNGSBERICHTE

DIE GEREGELTE LOGICAL CLOCK – DEFINITION, SIMULATION UND ANWENDUNG

EINE GLOBALE UHR FÜR TRACEBASIERTES SOFTWARE-MONITORING PARALLELER ANWENDUNGEN IN WORKSTATION CLUSTERN

Rolf Rabenseifner

E-Mail:
rabenseifner@rus.uni-stuttgart.de

ISSN 0941-4665 Mai 1996 RUS - 30

Rechenzentrum der Universität Stuttgart
Allmandring 30
D-70550 Stuttgart

Eine elektronische Version dieser Veröffentlichung und weitere Informationen finden Sie auf unserem WWW-Server unter folgender URL

`http://www.uni-stuttgart.de/People/rabenseifner/log_clock_rus30.html`

Abstract

Event tracing and monitoring of parallel applications are difficult if each processor has its own unsynchronized clock. A survey is given on several strategies to generate a global time, and their limits are discussed.

The controlled logical clock is presented. It is a new method based on Lamport's logical clock and provides a method to modify inexact timestamps of tracefiles. The new timestamps guarantee the clock condition, i.e. that the receive event of a message has a later timestamp than the send event. The corrected timestamps can also be used for performance measurements with pairs of events in different processes.

The controlled logical clock is motivated and it is analyzed in detail by computer simulations. No additional protocol overhead is needed for the new method while tracing the application. It can be implemented as a filter for tracefiles or it can be integrated into monitor and debug tools for parallel applications.

Das Aufzeichnen und Darstellen paralleler Anwendungen ist schwierig, wenn jeder Prozessor eine eigene Uhr besitzt und diese Uhren nicht synchronisiert sind. Mehrere Strategien zur Bildung einer globalen Uhrzeit werden in einem Überblick dargestellt, und ihre Grenzen werden aufgezeigt.

Die geregelte logical Clock, eine neue Methode auf der Basis von Lamports Logical Clock, wird vorgestellt. Ungenaue Zeitstempel aus Tracefiles werden derart modifiziert, daß sie einerseits die Clock Condition erfüllen, d.h. daß der Empfang einer Message einen späteren Zeitstempel als das zugehörige Sende-Ereignis besitzt, und daß andererseits Performancemessungen möglich werden, bei denen die Events in verschiedenen Prozessen liegen. Die neue Methode wird motiviert und anschließend eingehend mittels Computer-Simulationen analysiert.

Bei der Erstellung eines Tracefiles ist kein zusätzlicher Protokollaufwand nötig. Die geregelte logical Clock kann als Filter für Tracefiles implementiert werden. Sie kann aber auch in Monitor- und Debug-Tools integriert werden.

Keywords: Logical time, global time, clock, monitoring, debugging, clock synchronization, causality, distributed computing, parallel computing.

CR Kategorien:

D.1.3 [Concurrent Programming]	Distributed and parallel programming
D.2.5 [Testing and Debugging]	Monitors, Tracing
D.2.6 [Programming Environments]	
D.2.9 [Management]	Time estimation
D.4.8 [Performance]	Monitors, Measurements

Inhalt

1 Einleitung

Eine globale Uhrzeit ermöglicht eine optimale Darstellung von lokalen Ereignissen und Send-Receive-Paaren durch Monitortools zur Analyse paralleler und verteilter Anwendungen. Auf Systemen, die keine globale Uhr besitzen, gibt es verschiedene Strategien, eine solche anzunähern. Dieser Bericht stellt hierzu ein neues Verfahren vor – die geregelte logical Clock. Sie wurde speziell für Workstationcluster oder vergleichbare Systeme entwickelt und kann als Filter für Tracefiles eingesetzt werden. Hierbei werden die für die Zeitstempel eingesetzten Prozessoruhren nachträglich besser synchronisiert, wenn der Tracefile Rückwärtsbezüge enthält. Rückwärtsbezüge sind hierbei Send-Receive-Paare, bei denen das Send-Event einen späteren Zeitstempel als das Receive-Event besitzt. Die geregelte logical Clock modifiziert die Zeitstempel so, daß sie die Clock Condition [Lam78] erfüllen, d.h. ein Send-Event hat dann grundsätzlich einen früheren Zeitstempel als das zugehörige Receive-Event. Ein weiteres Einsatzgebiet sind Debugger. Außer der Erstellung der Zeitstempel und dem Zusammenführen der Traceinformation beim Monitor- oder Debuggingtool bedarf es keines zusätzlichen Protokollaufwands, wie z. B. der Austausch der Zeitstempel beim Messagetransfer selbst, da diese Tools eine vollständige Sicht der Anwendung besitzen.

Die geregelte logical Clock setzt voraus, daß mittels einer vorgeschalteten Synchronisation die Uhrenabweichungen auf ca. 2 ms begrenzt werden. Dies kann durch die von den meisten Tools angewandte explizite Synchronisation am Anfang und Ende der Anwendung [MT95] und anschließendem linearem Ausgleich, aber auch mittels resourcensparender kontinuierlicher Uhrensynchronisation [Mil92], erreicht werden.

Die geregelte logical Clock läßt die Zeitstempel der System Clocks C_i unverändert, solange die Zeitstempel von Receive-Events später als die der zugehörigen Send-Events zuzüglich der minimalen Übertragungszeit sind. Die geregelte logical Clock LC_i ist eine Modifikation von Lamports stückweise differenzierbarer logical Clock mit $dLC_i(t)/dt := dC_i(t)/dt$, wie sie auch in [Jez89a] beschrieben ist, doch im Gegensatz zu dieser ist bei der geregelten logical Clock die Abweichung $LC-C$ beschränkt. Sie ist damit zur Darstellung von Causal-Bezügen geeignet, als auch zur Messung von durchschnittlichen Zeitdifferenzen zwischen Ereignissen in verschiedenen Prozessen. Für durchschnittliche Zeitdifferenzen innerhalb eines Prozesses ist zwar die direkte Zeitmessung geeigneter, die Zeitstempel der geregelten logical Clock können aber trotzdem, z.B. für Vergleiche mit Transferzeiten, eingesetzt werden, da sie in der Regel höchstens 5 % abweichen, und die meisten Monitortools nur eine einheitliche Zeitskala unterstützen.

Kap. 2 gibt einen Überblick über grundlegende Arbeiten in diesem Gebiet, Kap. 3 bis 5 motivieren und definieren die einfache und die darauf aufbauende geregelte logical Clock. Diese wird in Kap. 6 mit simulierten Tracedaten analysiert und bewertet. Ihre Anwendung wird in Kap. 7 erörtert und in Kap. 8 werden die Ergebnisse zusammengefaßt.

2 Logical Clock und Synchronisation

Bei der Darstellung von Events zum Debugging und für Performancemessungen gibt es unterschiedliche Ansätze zur Synchronisation der lokalen Uhren. Lamport's diskrete Logical Clock [Lam78] kann direkt zum Monitoring eingesetzt werden [CHK92, ZEA93]. Raynal [Ray87] beschreibt hierzu ein Verfahren zur Begrenzung der Abweichung der Logical Clocks der Prozesse untereinander. Die Vektor Clock, eine Erweiterung von Fidge [Fid88, Fid89] und Mattern [Mat89], ermöglicht eine äquivalente Darstellung der durch die Send-Receive-Paare gegebenen Causalbeziehung mittels Monitortools [DW91, LK93, EK94]. Weitere Verbesserungen, z.B. zum Memory-Bedarf, werden in [CB91, MSV91, SK90, SES89, Che92, BDM+94, RS95] aufgezeigt. In [HW88] wird der Begriff der globalen Events eingeführt und in [PYS92] werden zusätzlich spontane Ereignisse, wie z.B. Kollisionen, berücksichtigt. In der Zusammenfassung [SM94] werden aber auch die Grenzen der diskreten logical Clock und der Vektor Clock beleuchtet.

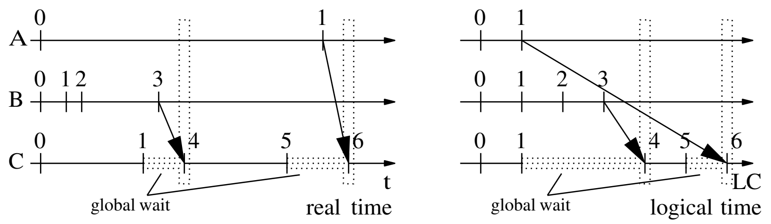


Abb. 2.1: Globales Wait dargestellt mittels Realtime und diskreter logical Clock

Abb. 2.1 zeigt eine weitere Beschränkung der Anwendbarkeit der logical Clock auf. Links ist ein Beispiel mit zwei globalen Waits über der realen Zeit aufgezeichnet – die Events sind aber mit ihrem logical Clock Zeitstempel gekennzeichnet. Rechts dagegen diente die Logical Clock als Zeitachse. Prozeß C wartet jeweils bis eine Message von A oder B kommt. Während des Wartens ist die Zeitachse des Prozesses punktiert dargestellt. Die rechte Darstellung ist nur bedingt geeignet, da sie den Anschein erweckt, daß Prozeß A schon zum Zeitpunkt 1 eine Message abgesandt hat, die beim ersten Wait nicht berücksichtigt wurde, obwohl Prozeß B erst viel später zum Zeitpunkt 3 seine Message abschickte. Dieses Beispiel zeigt, daß die logical Clock für die Darstellung von Monitorinformationen nicht ausreichend ist.

Eine Alternative hierzu ist, durch hinreichend genaue Synchronisierung der einzelnen Uhren eine Darstellung ohne Rückwärtsbezüge zu erreichen [Lam78, Hof93b, MT95]. Eine exakte Synchronisation am Anfang inklusive Driftbestimmung [Dun91, Dun94] oder besser am Anfang und Ende [MT95] mit einem linearen Ausgleich während der Dauer der Anwendung sind die in softwarebasierten Monitortools [RTV93] häufig angewandten Methoden. Für die genaue Synchronisation am Anfang und Ende sind deterministische [Sch87] und probabilistische [CF94, CF95] Verfahren geeignet.

Eine kontinuierliche Uhrensynchronisation mit geringer Resourcenbelastung, wie z. B. mit xntp [Mil92], ist normalerweise aufgrund der stark variierenden Messagedelays in einem lokalen Netz nicht ausreichend exakt.

Eine Alternative ist die tracebasierte Synchronisierung. Auf der Basis eines Tracefiles und der Prämisse, daß ein Receive-Event frühestens um die minimale Transferzeit nach dem Send-Event ankommen darf, werden die Uhrenabweichungen ermittelt. Duda [DHHB87] entwickelte hierzu zwei Verfahren mittels Regressionsanalyse, bzw. mittels der Bildung einer konvexen Hülle. Jezequel [Jez89a] erweiterte diese Verfahren für eine beliebige Prozessor-Topologie mittels eines *minimal spawning tree* Algorithmus und Hofmann [Hof93a] verbesserte diese durch eine vereinfachte Minimum/Maximum-Bildung und schlug eine Aufteilung der Gesamtzeit in Teilintervalle vor, damit auch Driftänderungen der Uhren bei lang laufenden Anwendungen ausgeglichen werden können. Babaoğlu und Drummond [BD87, DB93] zeigen, daß eine fast kostenlose Synchronisierung möglich ist, wenn die Anwendung hinreichend oft einen vollständigen Messageaustausch zwischen den Prozessen durchführt. Einen Überblick über weitere Arbeiten findet man in [YM93]. Die Grenzen dieser Methoden sind bestimmt durch die Varianz der Messagedelays, den nicht linearen Zusammenhang von Messagedelay und Messagelänge, sowie durch die teilweise einseitige Kommunikationstopologie von Anwendungen (z.B. Producer/Consumer Szenarien).

Die geregelte logical Clock ist eine neuartige Weiterentwicklung von Lamports Logical Clock in beiden Varianten, der diskreten und der mit $dLC_i(t)/dt := dC_i(t)/dt$ teilweise stetigen. Sie enthält aber auch Komponenten der tracebasierten Synchronisierung, da sie im Rahmen einer Traceauswertung oder eines Debuggingtools keinen zusätzlichen Protokolloverhead benötigt, sondern eine nachträgliche Synchronisation anhand der verhandenen Zeitstempel und Kommunikationsbeziehungen durchführt.

Im Gegensatz zur tracebasierten Synchronisation kann die geregelte logical Clock auch bei Uhren, deren Granularität größer als das minimale Messagedelay ist, angewandt werden.

Dafür bedarf die geregelte logical Clock einer vorgeschalteten Synchronisierung, die die Uhrenabweichung auf ca. das 4-fache des minimalen Messagedelays begrenzt.

3 Clock Condition

Problem: Für das Monitoring benötigt man Uhren, die einerseits für Performance-Messungen möglichst genau gehen sollten, und die andererseits für die Darstellung des Programmablaufs die nachfolgend definierte Clock Condition [Lam78] erfüllen sollten.

Definition 1: Es sei n die Anzahl der Prozesse,
e_i^j das j-te Ereignis im Prozeß i,
$E = \{e_i^j | i = 1..n, j = 0..j_{\max}(i)\}$ die Menge der Ereignisse, und
$M = \{(e_k^l, e_i^j) | e_k^l$ ist ein Sende-Ereignis und e_i^j ein zugehöriges Receive-Ereignis $\}$ ist die Menge der Send-Receive-Paare.
e_i^j ist ein internes Ereignis wenn es kein Sende- oder Receive-Ereignis ist.

Definition 2: Für zwei Ereignisse e_k^l, e_i^j gilt die Relation e_k^l *happened directly before* e_i^j, bzw. $e_k^l \tilde{\rightarrow} e_i^j$, wenn
(a) $(e_k^l, e_i^j) \in M$, oder (3.1)
(b) die Ereignisse im gleichen Prozeß aufeinanderfolgen, d.h. $k = i \wedge l = j - 1$. (3.2)

Definition 3: Die Relation *happened before*, bzw. $\rightarrow$, ist die transitive Hülle der Relation $\tilde{\rightarrow}$, d.h. die kleinste Relation, für die zusätzlich gilt:
(c) $e_k^l \rightarrow e_i^j \wedge e_i^j \rightarrow e_n^m \implies e_k^l \rightarrow e_n^m$ (3.3)

Definition 4: Eine Uhr $\mathcal{C} : E \mapsto \mathbb{R}$ erfüllt die *Clock Condition* wenn

$$\forall_{e_k^l \in E, e_i^j \in E} \; e_k^l \rightarrow e_i^j \implies \mathcal{C}(e_k^l) < \mathcal{C}(e_i^j) \tag{3.4}$$

Unter Vernachlässigung der Drift läßt sich folgender Satz einfach zeigen:

Satz 1 *Wenn die Abweichung der Prozessor-Uhren voneinander geringer ist als das kleinste Message-Delay, dann erfüllen die Prozessor-Uhren die Clock Condition.*

Eine Präzisierung für theoretisch stetige Uhren mit beschränktem Drift findet man in [Jez89b, MT95]. In [Hof93b] werden reale Uhren mit diskreten Sprüngen (Clock-Tick) untersucht.

Dieser Bericht untersucht den Fall, daß die Prämisse dieses Satzes nicht erfüllt ist, daß aber die Uhrendifferenzen kleiner als das ca. Vierfache des minimalen Messagedelays sind.

4 Die einfache logical Clock

Die einfache logical Clock ist eine Erweiterung von Lamports Logical Clock. Der Name *einfach* wurde zur Unterscheidung von der darauf aufbauenden *geregelten* logical Clock gewählt.

Doch zuerst wird auf der Basis eines Uhrenvergleichs am Anfang und Ende einer Anwendung, bzw. der Erstellung eines Tracefiles, eine Synchronisation mit linearem Ausgleich durchgeführt (SBA – sampling before and after [MT95]):

Algorithmus 1: Auf der Basis zweier Messungen zu den Zeitpunkten t_0 und t_e wird ein Driftausgleich durchgeführt:

$H_i(t)$ sei die Betriebssystem-Uhr des Prozesses i ($i = 1..n$),
t sei die reale Uhrzeit (UTC).
$T(t) := at + b$ sei die globale Zeit und (4.1)
$C_i(t) := a_i H_i(t) + b_i$ seien die korrigierten Prozessor-Uhren (4.2)
a, b, a_i und b_i sind dadurch definiert, daß (4.1) und (4.2) für $t - t_0$ und $t = t_e$ folgende Bedingungen erfüllen:

$$T(t_0) = T_0 \quad \text{und} \quad \underset{i}{\forall} C_{i=1..n}(t_0) = T_0 \tag{4.3}$$

$$T(t_e) = T_e \quad \text{und} \quad \underset{i}{\forall} C_{i=1..n}(t_e) = T_e \tag{4.4}$$

$$\text{mit} \quad T_0 := \frac{\sum_{i=1}^{n} H_i(t_0)}{n} \quad \text{und} \quad T_e := \frac{\sum_{i=1}^{n} H_i(t_e)}{n} \tag{4.5}$$

d.h. am Anfang und am Ende des Monitorings werden die Betriebssystem-Uhren gemittelt und daraus einerseits die globale Zeit T als lineare Funktion der realen Zeit und andererseits die Prozessor-Clocks C_i als lineare Funktion der Betriebssystem-Uhren gebildet.

Mittels (4.3) bis (4.5) sind alle a_i und b_i bestimmt. Da t_0 und t_e selbst nicht meßbar sind, können auch a und b nicht bestimmt werden. Dies stellt aber kein Problem dar, da sie im Folgenden nie benötigt werden.

Nun wird für jeden Prozeß eine logical Clock LC_i gebildet: sie bleibt (fast) stehen, solange $LC_i > C_i$. Ansonsten läuft sie wie C_i. Und beim Empfangen eines Events wird sie auf max ($LC_i, \mu + LC_k$ des Senders beim Absenden) vorgestellt, mit μ = minimales Messagedelay, ermittelbar bei der Synchronisation am Anfang und am Ende. Es handelt sich hierbei um eine Modifikation der von Lamport [Lam78] entwickelten Logical Clock. Sie wird im folgenden als **einfache logical Clock** bezeichnet und im nachfolgenden Kapitel zur geregelten logical Clock erweitert.

Algorithmus 2: Die exakte Definition der **einfachen logical Clock *LC*** ist:

$$LC_i(e_i^j) := \begin{cases} \max(LC_k(e_k^l) + \mu_{k,i},\ LC_i(e_i^{j-1}) + \delta_i,\ C_i(t(e_i^j))) & (4.6) \\ \quad \text{wenn} \underset{e_k^l}{\exists} (e_k^l, e_i^j) \in M & \\ \max(LC_i(e_i^{j-1}) + \delta_i,\ C_i(t(e_i^j))) \quad \text{andernfalls} & (4.7) \\ \text{wobei die Terme } LC_i(e_i^{j-1}) + \delta_i \text{ bei } j = 0 \text{ entfallen} & \end{cases}$$

mit
δ_i = minimaler Abstand zweier Events in Prozeß i, d.h. δ_i sind Konstanten für die gelten

$$\delta_i > 0 \quad \wedge \quad \underset{i,j}{\forall} T(t(e_i^j)) - T(t(e_i^{j-1})) \geq \delta_i \tag{4.8}$$

$\mu_{k,i}$ = minimales Delay einer Message von Prozeß k nach Prozeß i, d.h. $\mu_{k,i}$ sind Konstanten für die gelten

$$\mu_{k,i} > 0 \quad \wedge \quad \underset{(e_k^l, e_i^j) \in M}{\forall} T(t(e_i^j)) - T(t(e_k^l)) \geq \mu_{k,i} \tag{4.9}$$

Die globale logische Uhr ist definiert als

$$LC(e_i^j) := LC_i(e_i^j) \tag{4.10}$$

Satz 2 *Die einfache logical Clock LC erfüllt die* **clock condition***.*

Beweis: Der Algorithmus 2 genügt Lamports Regeln IR1 und IR2 in [Lam78] und erfüllt damit die Clock Condition.

□

Im folgenden wird der Fehler ausgehend von C_i modelliert:

$$\underset{e_i^j}{\forall} t_0 < t(e_i^j) < t_e \ \wedge \underset{t\,:\,t_0 < t < t_e}{\forall} -e^- \leq -e_i^- \leq C_i(t) - T(t) \leq e_i^+ \leq e^+ \tag{4.11}$$

d.h. die korrigierte Uhr gehe gegenüber der globalen Zeit um maximal e_i^- nach und um maximal e_i^+ vor.

Satz 3 *Wenn alle Uhren C_i nie mehr als e^+ vorgehen, dann geht auch die einfache logical Clock LC nie um mehr als e^+ vor, d.h.*

$$\underset{e_i^j}{\forall} C_i(t(e_i^j)) - T(t(e_i^j)) \leq e^+ \implies \underset{e_i^j}{\forall} LC(e_i^j) - T(t(e_i^j)) \leq e^+$$

Beweis: Angenommen es gibt ein (i, j) mit

$$LC(e_i^j) - T(t(e_i^j)) > e^+ \tag{4.12}$$

und o. B. d. A. sei e_i^j das früheste derartige Ereignis. (4.13)

Fall a) $LC(e_i^j) = LC_i(e_i^j)$ sei definiert mittels (4.6). Dann gilt

$$\begin{aligned} LC(e_i^j) - T(t(e_i^j)) &\overset{(4.10)}{=} LC_i(e_i^j) - T(t(e_i^j)) \\ &\overset{(4.6)}{=} \max(LC_k(e_k^l) + \mu_{k,i},\, LC_i(e_i^{j-1}) + \delta_i, \\ &\qquad C_i(t(e_i^j))) \;-\; T(t(e_i^j)) \end{aligned}$$

und (α) falls beim Maximum in (4.6) der erste Fall zutrifft:

$$\overset{(4.6,1.\text{Ausdruck})}{=} LC_k(e_k^l) + \mu_{k,i} - T(t(e_i^j)) \overset{(4.9)}{\leq} LC_k(e_k^l) - T(t(e_k^l)) \overset{(4.13)}{\leq} e^+$$

und (β) falls beim Maximum in (4.6) der zweite Fall zutrifft:

$$\overset{(4.6,2.\text{Ausdruck})}{=} LC_i(e_i^{j-1}) + \delta_i - T(t(e_i^j)) \overset{(4.8)}{\leq} LC_i(e_i^{j-1}) - T(t(e_i^{j-1})) \overset{(4.13)}{\leq} e^+$$

und (γ) falls beim Maximum in (4.6) der dritte Fall zutrifft:

$$\overset{(4.6,3.\text{Ausdruck})}{=} C_i(t(e_i^j)) - T(t(e_i^j)) \overset{\text{Prämisse}}{\leq} e^+$$

D.h. in allen drei Unterscheidungen ($\alpha - \gamma$) liegt ein Widerspruch zur Annahme (4.12) vor.

Fall b) $LC(e_i^j) = LC_i(e_i^j)$ sei definiert mittels (4.7). Dann gilt

$$\begin{aligned} LC(e_i^j) - T(t(e_i^j)) &\overset{(4.10)}{=} LC_i(e_i^j) - T(t(e_i^j)) \\ &\overset{(4.7)}{=} \max(LC_i(e_i^{j-1}) + \delta_i,\, C_i(t(e_i^j))) - T(t(e_i^j)) \\ &\overset{\text{Falla})(\beta),(\gamma)}{\leq} e^+ \end{aligned}$$

ebenfalls im Widerspruch zur Annahme (4.12).

Da in allen möglichen Fällen ein Widerspruch zur Annahme vorliegt, ist der Satz bewiesen.

□

Satz 4 *Wenn in einem Prozeß i die Uhr C_i nie mehr als e_i^- nachgeht, dann geht auch die einfache logical Clock LC in diesem Prozeß nie um mehr als e_i^- nach, d.h.*

$$\underset{i}{\forall}\, [\, \underset{j}{\forall}\, T(t(e_i^j)) - C(e_i^j) \leq e_i^- \implies \underset{j}{\forall}\, T(t(e_i^j)) - LC(e_i^j) \leq e_i^- \,]$$

Beweis:

$$\begin{aligned} & T(t(e_i^j)) - LC(e_i^j) \overset{(4.10)}{=} T(t(e_i^j) - LC_i(e_i^j)) \\ & \overset{(4.6,4.7)}{=} T(t(e_i^j)) - \left\{ \begin{array}{ll} \max(LC_k(e_k^l) + \mu_{k,i},\, LC_i(e_i^{j-1}) + \delta_i,\, C_i(t(e_i^j))) & \text{für ...} \\ \max(LC_i(e_i^{j-1}) + \delta_i,\, C_i(t(e_i^j))) & \text{andernfalls} \end{array} \right\} \\ & \overset{-\max(\text{...letzterTerm})}{\leq} T(t(e_i^j)) - \left\{ \begin{array}{l} C_i(t(e_i^j)) \\ C_i(t(e_i^j)) \end{array} \right\} \\ & \overset{\text{Prämisse}}{\leq} e_i^- \end{aligned}$$

□

Bemerkung 1 *Die beiden Sätze 3 und 4 sind nicht* symmetrisch *zueinander: Das Vorgehen der einfachen logical Clock LC ist nur global durch das Maximum des Vorgehens aller beteiligten Uhren begrenzt, während das Nachgehen von LC innerhalb eines jeden Prozesses durch das maximale Nachgehen der jeweiligen Prozessoruhr begrenzt ist.*

Satz 5 *Modellierung des Fehlers ausgehend von H_i:*

Wenn

$$H_i(t) = \alpha_i t + \beta_i + \varepsilon_i(t) \tag{4.14}$$

$$mit \quad |\varepsilon_i(t)| \leq e_i \tag{4.15}$$

dann gilt

$$|C_i(t) - T(t)| \leq 2a_i e_i$$

Beweis: Durch Einsetzen von (4.1), (4.2) und (4.14) in (4.3) und (4.4) und anschließender Elimination[1]von a und b bei festem i erhält man für $C_i(t) - T(t)$

$$C_i(t) - T(t) \overset{\left(\substack{4.1,\,4.2\\4.3,\,4.4\\4.14}\right)}{=} a_i \left(\varepsilon_i(t) - \frac{\varepsilon_i(t_0)(t_e - t) + \varepsilon_i(t_e)(t - t_0)}{t_e - t_0} \right) \tag{4.21}$$

Für den Betrag gilt wegen $\underset{x,y\in\mathbb{R}}{\forall} |x + y| \leq |x| + |y|$

$$\begin{aligned} |C_i(t) - T(t)| &\overset{(4.21)}{=} \left| a_i\varepsilon_i(t) - a_i \frac{\varepsilon_i(t_0)(t_e - t) + \varepsilon_i(t_e)(t - t_0)}{t_e - t_0} \right| \\ &\leq a_i|\varepsilon_i(t)| + a_i \frac{|\varepsilon_i(t_0)|(t_e - t) + |\varepsilon_i(t_e)|(t - t_0)}{t_e - t_0} \\ &\overset{(4.15)}{\leq} 2a_i e_i \end{aligned}$$

□

Wenn einer Betriebssystem-Uhr H_i eine konstante Quarzfrequenz zugrunde liegt, dann ist e_i der maximale Ablesefehler. Da jede Betriebssystem-Uhr eine „Tickbreite“ d_i besitzt, gilt $e_i \geq \frac{d_i}{2}$.

---

[1]

$$\begin{aligned} C_i(t) - T(t) &= a_i(\alpha_i t + \beta_i + \varepsilon_i(t)) + b_i - at - b & (4.16)\\ t = t_0 \ : \ 0 &= a_i(\alpha_i t_0 + \beta_i + \varepsilon_i(t_0)) + b_i - at_0 - b & (4.17)\\ t = t_e \ : \ 0 &= a_i(\alpha_i t_e + \beta_i + \varepsilon_i(t_e)) + b_i - at_e - b & (4.18) \end{aligned}$$

$$(4.18) - (4.17) \implies a = a_i\alpha_i + a_i \frac{\varepsilon_i(t_e) - \varepsilon_i(t_0)}{t_e - t_0} \tag{4.19}$$

$$t_e(4.17) - t_0(4.18) \implies b = a_i\beta_i + b_i + a_i \frac{t_e\varepsilon_i(t_0) - t_0\varepsilon_i(t_e)}{t_e - t_0} \tag{4.20}$$

(4.19) und (4.20) in (4.16) ergibt dann (4.21).

Für den Fall, daß die verschiedenen Prozessor-Uhren unterschiedlich lange Ticks haben, folgt aus der Bemerkung 1, daß man den durch die Tick-Länge induzierten Anteil am Fehler e_i^+ auf Null reduzieren sollte.

Im Gegensatz zu (4.15) ist hierzu der Fehler wie folgt zu modellieren:

$$
\begin{aligned}
H_i'(t) &= \alpha_i t + \beta_i + \varepsilon_i'(t) && (4.22)\\
&\quad \text{mit } -e_i^S \leq \varepsilon_i'(t) \leq +e_i^S \quad \text{für } t = t_0 \lor t = t_e && (4.23)\\
&\quad \text{und } -(e_i + \frac{d_i}{2}) \leq \varepsilon_i'(t) \leq +(e_i + \frac{d_i}{2}) \quad \text{für } t_0 < t < t_e && (4.24)
\end{aligned}
$$

Gl. (4.23) setzt voraus, daß der Synchronisationsalgorithmus die durchschnittliche Zeit erfaßt. e_i^S stellt den Fehler bei der Ermittlung der Uhrzeit von H_i' dar.

Gl. (4.24) modelliert, daß der Fehler bei der Zeitmessung zwei Anteile besitzt: d_i ist die Dauer eines Ticks und e_i entsteht durch nicht konstante Drift u.ä..

Wenn man für das Monitoring folgende Uhr einsetzt,

$$
\begin{aligned}
H_i(t) &= \alpha_i t + \beta_i + \varepsilon_i(t) \quad \text{wie (4.14) in Satz 5} \\
&\quad \text{mit } \varepsilon_i(t) = \varepsilon_i'(t) \quad \text{für } t = t_0 \lor t = t_e && (4.25)\\
&\quad \text{und } \varepsilon_i(t) = \varepsilon_i'(t) - \frac{d_i}{2} \quad \text{für } t_0 < t < t_e && (4.26)
\end{aligned}
$$

dann gilt

$$
\begin{aligned}
-e_i^S \leq \varepsilon_i(t) \leq +e_i^S \quad &\text{für} \quad t = t_0 \lor t = t_e && (4.27)\\
-(e_i + d_i) \leq \varepsilon_i(t) \leq +e_i \quad &\text{für} \quad t_0 < t < t_e && (4.28)
\end{aligned}
$$

Satz 6 *Wenn H_i gemäß (4.22) bis (4.28) modelliert ist, dann gilt*

$$\forall_{t_0 < t < t_e} \; - a_i(e_i + e_i^S + d_i) \leq C_i(t) - T(t) \leq a_i(e_i + e_i^S) \qquad (4.29)$$

Beweis: Da (4.14) aus Satz 5 übernommen wurde, gilt auch (4.21) aus dessen Beweis.
Die rechte Seite von (4.29) ergibt sich aus

$$
\begin{aligned}
C_i(t) - T(t) &\overset{(4.21)}{=} a_i \varepsilon_i(t) + a_i \frac{-\varepsilon_i(t_0)(t_e - t) - \varepsilon_i(t_e)(t - t_0)}{t_e - t_0} \\
&\leq a_i \max_{t_0 < t < t_e}(\varepsilon_i(t)) + a_i \max(-\varepsilon_i(t_0), \varepsilon_i(t_e)) \\
&\overset{\substack{(4.28)\text{rechts} \\ (4.27)\text{links}}}{\geq} a_i(e_i + e_i^S)
\end{aligned}
$$

Die linke Seite von (4.29) ergibt sich aus

$$
\begin{aligned}
C_i(t) - T(t) &\overset{(4.21)}{=} a_i \varepsilon_i(t) + a_i \frac{-\varepsilon_i(t_0)(t_e - t) - \varepsilon_i(t_e)(t - t_0)}{t_e - t_0} \\
&\geq a_i \min_{t_0 < t < t_e}(\varepsilon_i(t)) + a_i \min(-\varepsilon_i(t_0), \varepsilon_i(t_e)) \\
&\overset{\substack{(4.28)\text{links} \\ (4.27)\text{rechts}}}{\leq} a_i(-e_i - d_i - e_i^S)
\end{aligned}
$$

□

Satz 7 *Mit* $e_i^- := a_i(e_i + e_i^S + \mathbf{d_i})$ *und* $e^+ := \max_{i=1..n}(a_i(e_i + e_i^S))$ *gilt*

$$\forall_{e_i^j} - e_i^- \leq LC(e_i^j) - T(t(e_i^j)) \leq e^+ \tag{4.30}$$

Beweis: Dieser Satz folgt direkt aus den Sätzen 3 und 4, da e_i^- und e^+ gemäß Satz 6 deren Prämissen erfüllen.

□

5 Die geregelte logical Clock

Im Algorithmus 2 bewirkt der Term $LC_k(e_k^l) + \mu_{k,i}$ in (4.6) ggf. ein Vorstellen der logical Clock $LC(e_i^j)$ gegenüber der Uhr $C_i(t(e_i^j))$.
Bei nachfolgenden Ereignissen im gleichen Prozeß bewirkt der Term $LC_i(e_i^{j-1}) + \delta_i$ in (4.6) und (4.7), daß LC im Prinzip **stehenbleibt** (bzw. – genau gesagt – um eine kleine Zeitspanne δ_i vorgestellt wird), bis sie dann wieder *zurückgefallen* ist auf den Wert $C_i(t(e_i^j))$ (siehe Term $C_i(t(e_i^j))$ in (4.6) und (4.7)).

Damit die logical Clock gleichmäßiger gehen kann, bzw. um das abwechselnde Vorstellen und Stehenbleiben derselben zu verhindern, wird die einfache logical Clock aus Algorithmus 2 folgendermaßen erweitert:

Algorithmus 3: LC' bildet die Basis f'ur die geregelte logical Clock:

$$LC_i'(e_i^j) := \begin{cases} \max(LC_k'(e_k^l) + \mu_{k,i},\ LC_i'(e_i^{j-1}) + \delta_i, \\ \qquad LC_i'(e_i^{j-1}) + \gamma_i^j(C_i(t(e_i^j)) - C_i(t(e_i^{j-1}))), \\ \qquad C_i(t(e_i^j))) \quad \text{wenn } \underset{e_k^l}{\exists}\ (e_k^l, e_i^j) \in M & (5.1) \\ \max(LC_i'(e_i^{j-1}) + \delta_i,\ LC_i'(e_i^{j-1}) + \gamma_i^j(C_i(t(e_i^j)) - C_i(t(e_i^{j-1}))), \\ \qquad C_i(t(e_i^j))) \quad \text{andernfalls} & (5.2) \\ \text{wobei die Terme } LC_i(e_i^{j-1}) + \delta_i \text{ bei } j = 0 \text{ entfallen} \end{cases}$$

mit δ_i und $\mu_{k,i}$ wie in Alg. 2 und mit frei wählbaren $\gamma_i^j \in [0, 1]$.

Die globale logische Uhr wird wie in Alg. 2 (4.10) ebenfalls aus den logical Clocks der einzelnen Prozesse gebildet:

$$LC'(e_i^j) := LC_i'(e_i^j) \tag{5.3}$$

Der Algorithmus wird mit dem Regler Alg. 4 später vervollständigt.

Offensichtlich ist $LC_{\text{Alg.2}} = LC'_{\text{Alg.3}}$ für $\gamma_i^j \equiv 0$.

Für $\gamma_i^j \equiv 1$ entspricht LC' der Definition von Lamport [Lam78] für eine stückweise differenzierbare Logical Clock LC mit $d\,LC_i/dt = d\,C_i/dt$ für alle Zeitpunkte zwischen Receive-Events.

Während der Satz 4 über die Beschränktheit des Nachgehens ebenfalls für LC' gilt[1], kann man zeigen, daß der Satz 3 über die Beschränktheit des Vorgehens für LC' nicht gilt. Man kann sogar zu jeder Wahl eines Wertes e und einer beliebig kleinen positiven unteren Schranke für γ_i^j ein Beispiel konstruieren, indem für ein e_i^j die logical Clock LC' mehr als e vorgeht.

Beispiel 1: n Prozesse P_i haben lokale Uhren C_i mit Tick d, d.h. die Uhren bleiben d Sekunden stehen, um anschließend einen Sprung um d Sekunden zu machen. Die

[1]Der Beweis kann direkt übernommen werden

Laufzeit einer Message sei μ. Das Kommunikationsmuster der Anwendung sei folgendermaßen: P_1 sendet eine Message m_1 zu P_2. Sofort nach dem Empfangen sendet dieser die Message m_2 zu P_3, usw. bis P_{n-1} die Message m_{n-1} zu P_n schickt. Zwischen dem Empfangen und dem Absenden einer Message liege die Zeitspanne δ. Zufälligerweise sei der Sprung der Uhr C_i nachdem P_i die Message m_{i-1} von P_{i-1} empfangen hat und bevor P_i die Message m_i nach P_{i+1} schickt. Anschliessend werden die Messages $b_n, b_{n-1}, ...b_2$ von P_n nach P_{n-1}, von P_{n-1} nach P_{n-2}, usw. bis von P_2 nach P_1 gesendet. Während dieser Phase finden keine Uhrensprünge statt. Damit ein derartiger Kommunikationszyklus innerhalb eines Ticks stattfinden kann, muß $2(n-1)(\mu+\delta) \leq d$ sein.

Das Beispiel ist in Abb. 5.1 dargestellt. Zum besseren Verständnis wurden einige interne Events zusätzlich hinzugefügt. In der Rechnung wurden sie aber nicht berücksichtigt. Nach m Zyklen beträgt die Abweichung $A(m) = m \max([(n\gamma d + 2(n-1)\mu + (n-2)\delta) - d], 0)$.

Für $\gamma > \gamma_{min}$ mit $\gamma_{min} := \frac{d-2(n-1)\mu-(n-2)\delta}{\boldsymbol{n} \cdot d}$ kann die Abweichung beliebig groß werden. Es gilt dann $A(m) = m\boldsymbol{n}d(\gamma - \gamma_{min})$.[2]

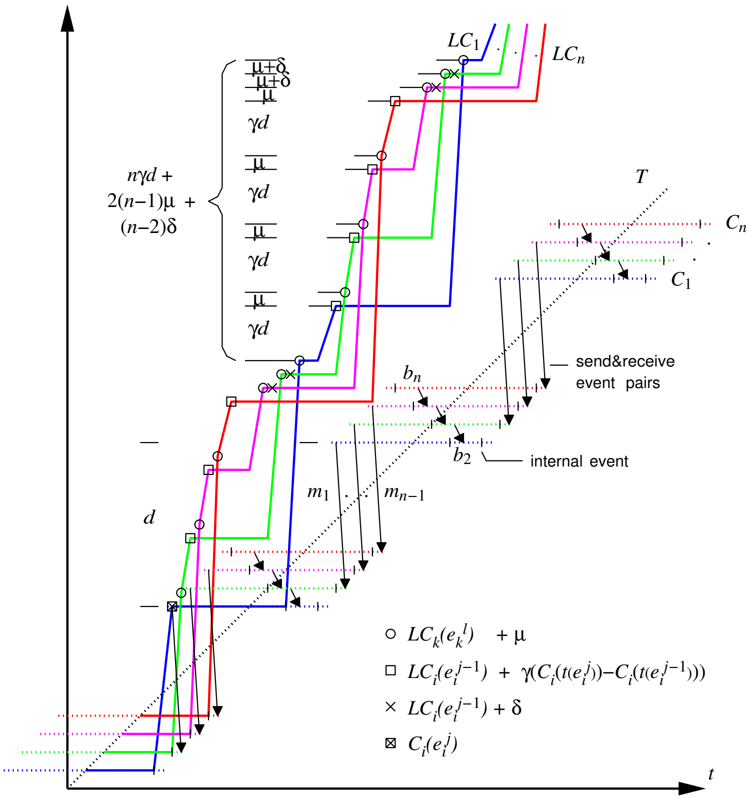


Abb. 5.1: Beispiel mit *unbeschränktem* LC'

Zahlenbeispiele:

$d = 1000\mu s$, $\mu = 250\mu s$, $\delta = 4\mu s$, $n = 2$

$\Rightarrow \gamma_{min} = 0.250$ und $A(m) = m \cdot 2ms \cdot (\gamma - \gamma_{min})$

$d = 10000\mu s$, $\mu = 250\mu s$, $\delta = 4\mu s$, $n = 20$

[2] Die wesentlichen Unterschiede zum nachfolgenden Beispiel sind hervorgehoben.

$\Rightarrow \gamma_{min} = 0.00214$ und $A(m) = m \cdot 200ms \cdot (\gamma - \gamma_{min})$

$d = 10000\mu s,\ \mu = 250\mu s,\ \delta = 4\mu s,\ n = \frac{n_{max}}{2} = 10$

$\Rightarrow \gamma_{min} = \mathbf{0.05}468$ und $A(m) = m \cdot 100ms \cdot (\gamma - \gamma_{min})$

Beispiel 2: Das folgende Beispiel soll zeigen, daß selbst bei einer Tick-Dauer von $d = 10000\mu$s für $\gamma = 0.5$ der Fehler von LC' beschränkt bleiben kann.

Dieses Beispiel ist wie Beispiel 1 nur, daß die Uhren der Prozesse P_i jeweils nach dem Senden der Message m_i und vor dem Empfangen von b_{i-1} um einen Tick weitergehen.

In diesem Fall – siehe Abb. 5.2 – ist die Abweichung $A(m) = m \max([(\gamma d + 2(n-1)\mu + (2n-3)\delta) - d], 0)$, d.h. für $\gamma > \gamma_{min} := \frac{d-2(n-1)\mu+(2n-3)\delta}{d}$ gilt $A(m) = md(\gamma - \gamma_{min})$.[3]

Zahlenbeispiel: (s.a. letztes Zahlenbeispiel im vorherigen Beispiel)

$d = 10000\mu s,\ \mu = 250\mu s,\ \delta = 4\mu s,\ n = \frac{n_{max}}{2} = 10$

$\Rightarrow \gamma_{min} = \mathbf{0.5}432$ und $A(m) = m \cdot 10ms \cdot (\gamma - \gamma_{min})$

bzw. für $\gamma < \gamma_{min}$ ist $A(m) = 0$.

Eine normale Anwendung[4] hat in den meisten Fällen[5] ein $\gamma_{min} > 0.5$.

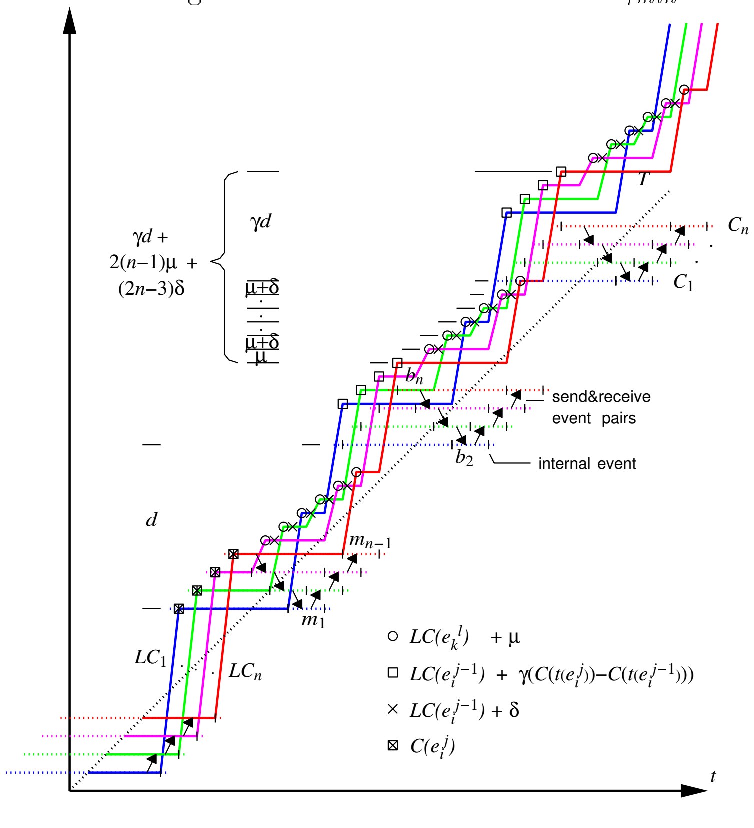


Abb. 5.2: Beispiel mit geringerem Fehlerwachstum

Da aber normalerweise γ-Werte im Bereich nahe 1 wünschenswert sind, ist es sinnvoll, $\gamma_i^j \in [0,1]$ über einen Regelkreis zu steuern. Abb. 5.3 zeigt seinen prinzipiellen Aufbau.

[3]Die Unterschiede zum vorhergehenden Beispiel sind dort in den Formeln für γ_{min} und $A(m)$ hervorgehoben.

[4]die nicht nur kommuniziert, sondern auch etwas berechnet

[5]bezogen auf die Zufälligkeiten der Uhrenabweichungen

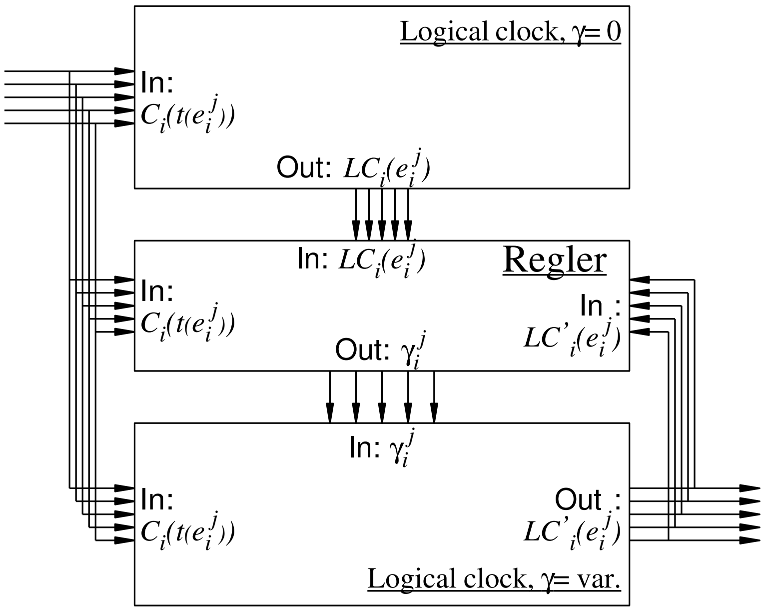


Abb. 5.3: Regelkreis für γ_i^j

Es ist das Ziel des Reglers, die Abweichung von LC' gegenüber T, d.h. den Ausgangsfehler $LC' - T$ zu begrenzen. Da $T(t(e_i^j))$ unbekannt ist, muß der Regler den Ausgangsfehler indirekt abschätzen. Für den Eingangsfehler $C - T$ gilt gleiches. Hier sind die Sätze 3 und 4 hilfreich:

$$\forall_{e_i^j} \; -e^- \le -e_i^- \le \; C_i(t(e_i^j)) - T(t(e_i^j)) \; \le e_i^+ \le e^+ \tag{4.11}$$

$$\Downarrow$$

$$\forall_{e_i^j} \; -e_i^- \le \; LC(e_i^j) - T(t(e_i^j)) \; \le e^+ \qquad \text{(Satz 3, 4)}$$

Statt $LC' - T$ und $C - T$ wird $LC' - C$ und $LC - C$ verglichen:

$$\begin{aligned} LC' - C &= (LC' - T) - C - T \\ LC - C &= (LC - T) - C - T \end{aligned}$$

Mit (4.11) und Satz 3 und 4 gilt:

$$-e_i^+ - e_i^- \le \underbrace{LC_i - C_i}_{\text{meßbar}} \le e^+ + e_i^-$$

und mit der Def. von e_i^- und e_i^+ (4.11):

$$-e_i^- \le \underbrace{(LC_i' - T)}_{\substack{\text{vom Regler}\\ \text{zu begrenzen}}} - \underbrace{(LC_i' - C_i)}_{\text{meßbar}} \le e_i^+$$

d.h. wenn der Regler $LC' - C$ zwischen $-e_i'^-$ und $+e_i'^+$ begrenzt, d.h. wenn

$$-e_i'^- \le LC' - C \le +e_i'^+$$

dann gilt

$$-\boldsymbol{e}_i^- - \boldsymbol{e}_i'^- \leq -e_i^- + (LC_i' - C_i) \leq \boldsymbol{LC}_i' - \boldsymbol{T} \leq e_i^+ + (LC_i' - C_i) \leq \boldsymbol{e}_i^+ + \boldsymbol{e}_i'^+$$

Diese Überlegungen motivieren folgenden **Regler**-Algorithmus:

Algorithmus 4: Die logischen Uhren in den Algorithmen 2 und 3 werden hierzu schrittweise berechnet. In jedem Schritt s werden für alle diejenigen Prozesse $i \in P_s$ neue Werte für $LC_i(e_i^j)$ und $LC_i'(e_i^j)$ berechnet[6], für die die Ausgangswerte schon berechnet sind (d.h. $i \notin P_s$, wenn $\exists_{e_k^l} (e_k^l, e_i^j) \in M$ und $LC_k(e_k^l)$ bzw. $LC_k'(e_k^l)$ noch nicht in einem früheren Schritt berechnet worden sind).

Nach jedem Berechnungsschritt werden die Regelgrößen γ_i^j für $i \in P_s$ mit folgendem Algorithmus neu berechnet:

$$\begin{aligned}
D_{i,0} &:= q_{init} && (5.4)\\
D'_{i,0} &:= q_{init} && (5.5)\\
D_{i,s} &:= \max(LC_i(e_i^j) - C_i(t(e_i^j)), q_{factor}(D_{i,s-1} - q_{min}) + q_{min}) && (5.6)\\
D'_{i,s} &:= \max(LC_i'(e_i^j) - C_i(t(e_i^j)), q_{factor}(D'_{i,s-1} - q_{min}) + q_{min}) && (5.7)\\
\gamma_i^1 &:= \gamma_{max} && (5.8)
\end{aligned}$$

$$\gamma_i^{j+1} := \begin{cases} \gamma_i^j \cdot \gamma_{degress} & \text{wenn } D'_{i,s} > l_{upper} \cdot D_{i,s} \quad (5.9)\\ \min(\gamma_i^j / \gamma_{degress}, \gamma_{max}) & \text{wenn } D'_{i,s} < l_{lower} \cdot D_{i,s} \quad (5.10)\\ \gamma_i^j & \text{sonst} \quad (5.11) \end{cases}$$

Erläuterungen:

- In jedem Schritt wird für möglichst viele Prozesse ein neuer Logical Clock Wert berechnet.
- Der hier vorgeschlagene Regelmechanismus berechnet γ_i^{j+1} nur auf der Basis der zum Prozeß i gehörenden Daten.
- Dadurch ist die Berechnung der Logical Clock Werte teilweise und der Regelmechanismus vollständig parallelisierbar.
- $q_{init}, q_{min}, q_{factor}, \gamma_{max}, \gamma_{degress}, l_{upper}$ und l_{lower} sind Parameter, mit denen der Regler eingestellt werden muß.
- $D_{i,s}$ ist ein Maß für die Abweichnug der Logical Clocks LC_i von den Betriebssystem-Clocks C_i. $D_{i,s}$ berechnet das Maximum von $LC - C$, wobei ältere Werte langsam vergessen werden. $q_{factor} \in (0,1)$ ist der *Vergeßfaktor*, s. (5.6).
- q_{min} in (5.6) ist eine untere Schranke für $D_{i,s}$. Sie besagt, welche Uhrenfehler noch als tolerabel betrachtet werden.

[6] j sei eine Abkürzung für $j_{i,s}$ und $j_{i,s}$ sei der Index des Ereignisses, dessen Logical Clock im Schritt s im Prozeß i berechnet wird.

- q_{min} sollte auch als Startwert q_{init} in (5.4) gewählt werden.
- $D'_{i,s}$ ist analog zu $D_{i,s}$ definiert. Es ist ein Maß für die Abweichnug der geregelten logical Clocks LC'_i von den Betriebssystem-Clocks C_i und $D'_{i,s}$ stellt damit eine Maximum-Bildung von $LC' - C$ dar (s. (5.5) und (5.7)).
- Der Regler versucht nun mit (5.9) die Abweichung $D'_{i,s}$ der geregelten logical Clock zu begrenzen, sobald sie das obere Limit $l_{upper} \cdot D_{i,s}$ übersteigt. Durch die Reduktion von γ_i^j wird das Verhalten der geregelten logical Clock LC' schrittweise dem Verhalten der einfachen logical Clock LC angepaßt.
- Sobald $D'_{i,s}$ wieder unter das untere Limit $l_{lower} \cdot D_{i,s}$ sinkt, werden die γ_i^j wieder schrittweise mit (5.10) erhöht, wobei γ_{max} eine obere Schranke ist.
- $1 - \gamma_{max}$ besagt, wie starkt die geregelte logical Clock gebremst wird, wenn sie ein Vorgehen wieder ausgleichen will.
- Folgende Werte werden aufgrund der im Kapitel 6 beschriebenen Ergebnisse für ein Ethernet- oder FDDI-basiertes Cluster empfohlen:

$$\begin{aligned}
q_{init} &= 250\mu\text{s} \\
q_{min} &= 250\mu\text{s} \\
q_{factor} &= 0.9 \\
\gamma_{max} &= 0.95 \\
\gamma_{degress} &= 0.9 \\
l_{upper} &= 2.0 \\
l_{lower} &= 1.8
\end{aligned}$$

6 Simulation der geregelten logical Clock

Die Effektivität der in den Algorithmen 3 und 4 definierten geregelten logical Clock wird in diesem Kapitel anhand von Computersimulationen untersucht.

Die Komponenten dieser Computersimulationen sind

- der simulierte Ablauf einer parallelisierten FE-Rechnung auf einem regulären Gitter erzeugt eine Ereignismenge $E = \{e_i^j\}$ an den Zeitpunkten $t(e_i^j)$,
- simulierte Betriebssystem-Uhren mit vorgegebenem Fehlverhalten erzeugen Abweichungen zwischen $C(t(e_i^j))$ und $t(e_i^j)$,
- die geregelte logical Clock errechnet zu $C(t(e_i^j))$ die Werte $LC'(e_i^j)$ und $LC(e_i^j)$,
- und ein Analysemodul bewertet die geregelte logical Clock.

6.1 Die Ereignismenge

Die Ereignismenge wurde durch den simulierten Ablauf einer parallelisierten FE-Rechnung gewonnen.

Die FE-Rechnung wird auf einem rechteckigen Gitter mit $n_1 \times n_2$ Prozessen ausgeführt. Jeder Prozeß hat die Aufgabe in jedem Simulationsschritt eine größere Anzahl finiter Elemente zu berechnen.

Der Ablauf in jedem Prozeß ist folgender

- Berechnung der finiten Elemente an den Rändern zu den Nachbarn.
- Versenden dieser neuen Ergebnisse zu den Nachbarn.
- Berechnen der restlichen finiten Elemente.
- Empfangen der neuen Nachbarwerte.

Bei diesem Ablauf werden pro Zyklus folgende Events generiert:

do $i_1 = 1, exa3\_repeats$
    do $i_2 = 1, rounds$
        execution 1 – finite elements on the borders
            internal event (tag=4)
            $T = T +$ value between $ex1\_time\_min..ex1\_time\_max$
            internal event (tag=1)
        sending messages to all neighbours
            send event with $T_{receiveable} = T + delay\_min..delay\_max$ (tag=2)
            T = T + send_delay
        execution 2 – finite elements on the borders
            internal event (tag=2)
            $T = T +$ value between $ex2\_time\_min..ex2\_time\_max$

```
            internal event (tag=3)
        receiving from all neighbours
            T = T + rcv_delay
            wait for event
            receive event (tag=4)
    enddo
enddo
```

Hierbei wurden unterschiedliche Rechenzeiten dadurch simuliert, daß bei jedem Rechenschritt die zugehörige Rechenzeit mittels eines gleichverteilten Zufallsgenerators in den vorgegebenen Intervallen $[ex1\_time\_min, ex1\_time\_max)$ und $[ex2\_time\_min, ex2\_time\_max)$ neu festgelegt wurde.

Die unterschiedlichen Transferzeiten für Messages wurden ebenfalls mittels eines gleichverteilten Zufallsgenerators im Intervall $[delay\_min, delay\_max)$ festgelegt. Es wurde angenommen, daß der Art der Verteilung nur eine untergeordnete Bedeutung bei der Bewertung der geregelten logical Clock zukommt.

Das Versenden der Messages wurde als asynchron (und ggf. gepuffert) betrachtet. Für das Initieren des Sendevorgangs bzw. des Empfangens einer Message wurde der konstante Rechenzeitbedarf $send\_delay$, bzw. rcv_delay angesetzt.

Der Wert $rounds$ gibt an, wieviele Iterationen durchgeführt werden, bei denen der Zufallsgenerator für die Rechenzeiten und Messagedelays immer neue Werte bestimmt und damit immer neue Konstellationen erzeugt, wann welcher Prozeß auf welchen warten muß.

Der Wert $exa3\_repeats$ gibt an, wie oft die bis dahin definierte Ereignisreihe identisch, aber um die Dauer dieser Ereignisreihe versetzt, wiederholt wird.

Es handelt sich bei diesem Experiment um ein typisches Szenarium, bei dem die Messagedelays durch parallel laufende Rechnungen teilweise unschädlich gemacht werden.

Sofern nichts anderes angegeben, wurden folgende Werte bei den Experimenten benutzt:

$$\begin{aligned} n_1 &= 4 \\ n_2 &= 5 \\ rounds &= 100 \\ exa3\_repeats &= 87 \\ ex1\_time\_min &= 10\mu\text{s} \\ ex1\_time\_max &= 50\mu\text{s} \\ ex2\_time\_min &= 100\mu\text{s} \\ ex2\_time\_max &= 1000\mu\text{s} \\ delay\_min &= 250\mu\text{s} \\ delay\_max &= 1000\mu\text{s} \\ send\_delay &= 50\mu\text{s} \\ rcv\_delay &= 50\mu\text{s} \end{aligned}$$

Dies führt zu einer Gesamtsimulationszeit von $\tau_{ges} = 10\text{s}$.

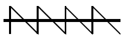
6.2 Simulierte Uhrenfehler

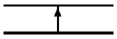
Das im vorherigen Kapitel beschriebene Experiment setzt zu jedem Ereignis e_i^j die Uhrzeit $t(e_i^j)$ fest.

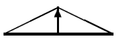
Bei der Simulation der Uhrenfehler verzichten wir auf eine lineare Abweichung zwischen T und t, da diese Abweichung durch den Algorithmus 1 auf S. 9 wieder vollständig eliminiert werden würde. Daher wird $T(t) := t$ gesetzt und es gilt $C_i(t) = H_i(t)$.

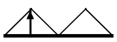
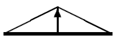
Im folgenden werden daher verschiedene Modelle für fehlerbehaftete Prozessor-Clocks $C_i(t)$ beschrieben. In den Pictogrammen ist die Abweichung $C_i(t) - t$ über der Zeit t aufgetragen.

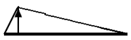
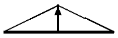
 Die Uhr geht exakt, d.h. $C_i(t) = t$

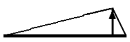
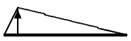
 Die Uhr besitzt eine grobe Schrittweite, d.h. $C_i(t) = \lfloor (t - \tau_b + \frac{\tau_l}{2})/\tau_l \rfloor \tau_l + \tau_s$ mit der Schrittweite τ_l und dem Synchronisationspunkt[1] τ_sr.

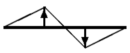
 Die Uhr geht gleichbleibend vor, d.h. $C_i(t) = t + \Delta$.

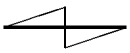
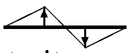
 Das Vorgehen der Uhr wächst linear von 0 auf Δ während der ersten Hälfte der Gesamtzeit τ_{ges}, um anschließend wieder linear auf 0 zu sinken, d.h.
$C_i(t) = (1 + \frac{\Delta}{0.5\tau_{ges}})t$ für $t \in [0, 0.5\tau_{ges}]$ und
$C_i(t) = (1 - \frac{\Delta}{0.5\tau_{ges}})t + 2\Delta$ für $t \in [0.5\tau_{ges}, \tau_{ges}]$. D.h. im ersten Zeitintervall besitzt die Uhr einen konstanten positiven Drift von $\frac{\Delta}{0.5\tau_{ges}}$ und im zweiten Intervall einen konstanten negativen Drift.

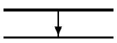
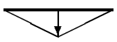
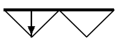
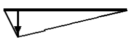
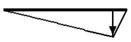
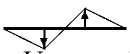
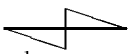
 Wie , wobei die Gesamtzeit in 4 Teile aufgeteilt ist und zweimal ein Vorgehen von Δ erreicht wird.

 Wie , nur daß das maximale Vorgehen innerhalb von 10% der Gesamtzeit erreicht wird und das Absinken auf 0 innerhalb der restlichen 90% geschieht.

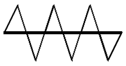
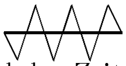
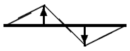
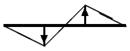
 Wie , nur daß das Ansteigen innerhalb 90% der Gesamtzeit erreicht wird, während das Absinken dann in den restlichen 10% geschieht.

 Im ersten Drittel der Gesamtzeit geht die Uhr linear steigend von 0 bis Δ vor, im nächsten Drittel läuft sie so verlangsamt, daß sie am Ende dieses Drittels um Δ nachgeht, und im letzten Drittel läuft sie wieder so beschleinigt, daß sie am Ende wieder exakt geht.

 Wie , nur daß das mittlere Zeitintervall statt eines Drittels der Gesamtzeit nur 2Δ lang ist. Während dieser mittleren Phase bleibt die Uhr stehen. Die Dauer der beschleunigten Phasen am Anfang und Ende ist jeweils die Hälfte der Gesamtdauer abzüglich Δ.

, , , , , , und  entsprechen obigen Modellen, wobei die Rolle von Vor- und Nachgehen vertauscht sind

[1]Für $t = \tau_b + n\tau_l$ mit $n = 1, 2, ...$ gilt $C_i(t) = t$

 und  sind drei mal  bzw.  in jeweils einem Drittel der Zeit.

Die verschiedenen Fehlermethoden werden bei der Analyse in Kap. 6.5 auf Seite 27 in unterschiedlichen Kombinationen für die einzelnen Prozessoren eingesetzt.

Soweit nichts anderes angegeben ist, gilt immer

$$\begin{array}{rclrcl} \Delta & = & 1000\mu\text{s} & \tau_l & = & 1000\mu\text{s} \\ \tau_{ges} & = & 10\text{s} & \tau_b & = & -(i-1)\cdot 1000\mu\text{s}/n \end{array}$$

Die hierdurch auftretenden Driftschwankungen liegen zwischen $4 \cdot 10^{-4}$ und $11 \cdot 10^{-4}$. Die durch normale Uhrenquarze verursachten Driftschwankungen liegen dagegen im Bereich $10^{-6}...10^{-5}$. Im Kap. 6.5 wird ebenfalls gezeigt, daß die mit der geregelten logical Clock erzielbaren Ergebnisse weitgehend von τ_{ges} unabhängig sind und nur von den absoluten Abweichungen Δ abhängen, d.h. bei Simulationszeiten von $\tau_{ges} = 1000$s treten die dort gezeigten Phänomene auch im Zusammenhang mit den (temperaturbedingten) normalen Driftschwankungen der Uhrenquarze auf.

Diese Fehlermodelle sollen aber auch das Verhalten extern (z.B. mittels xntp [Mil92]) synchronisierter Uhren berücksichtigen, die durch additive Driftschwankungen mit UTC [DS93] synchronisiert werden. Die hierbei auftretenden Abweichungen von UTC liegen in der Größenordnung von 1 ms.

6.3 Das logical Clock Modul

Es handelt sich hierbei um eine Implementierung der Algorithmen 2, 3 und 4.

Diese Implementierung benötigte ca. sizeof(void$*$) $\cdot\, n^2 + 100 \cdot n$ Bytes Arbeitsspeicher und ca. $\max(6.2/n^{0.0726}, 2.85n^{0.1911})\mu$s Rechenzeit pro Event auf einem R8000-Prozessor, d.h. zwischen 5 und 11 μs bei 1 bis 1000 ereignisproduzierenden Prozessen.

6.4 Analysemethoden

Der Vorteil, daß im Experiment nur schwer oder gar nicht zugängliche Observable in einer Computersimulation problemlos beobachtet werden können, wird auch hier bei der Analyse der geregelten logical Clock ausgenutzt:

Die globale Zeit t kann als Basis für die Bewertung der geregelten logical Clock LC' im Vergleich zur einfachen logical Clock LC und zur Prozessor-Uhr C dienen.

Observable	Bedeutung und Kriterium	Einheit
$\overline{\mathcal{V}_{LC'}} = \overline{\overline{LC' - T}^{+}}$	durchschnittliches Vorgehen der geregelten logical Clock, d.h. $E(\max(LC'(e_i^j) - t(e_i^j), 0)$	$[\mu s]$
$\overline{\mathcal{V}_{LC}} = \overline{\overline{LC - T}^{+}}$	durchschnittliches Vorgehen der einfachen logical Clock, d.h. $E(\max(LC(e_i^j) - t(e_i^j), 0)$	$[\mu s]$
$\overline{\mathcal{V}_{C}} = \overline{\overline{C - T}^{+}}$	durchschnittliches Vorgehen der Prozessor-Clock, d.h. $E(\max(C(t(e_i^j)) - t(e_i^j), 0)$	$[\mu s]$
$\frac{\overline{\mathcal{V}_{LC'}}}{\overline{\mathcal{V}_{LC}}} = \frac{\overline{\overline{LC'-T}^{+}}}{\overline{\overline{LC-T}^{+}}}$	Vorgehen der geregelten im Vergleich zur einfachen logical Clock; aufgrund der Definition der geregelten logical Clock muß diese in diesem Parameter schlechter sein als die einfache logical Clock, d.h. der Quotient ist größer als 1. **Kriterium: möglichst kleiner als 2**	[1]
$\overline{\mathcal{N}_{LC'}} = \overline{\overline{LC' - T}^{-}}$	durchschnittliches Nachgehen der geregelten logical Clock, d.h. $E(\min(LC'(e_i^j) - t(e_i^j), 0)$	$[\mu s]$
$\overline{\mathcal{N}_{LC}} = \overline{\overline{LC - T}^{-}}$	durchschnittliches Nachgehen der einfachen logical Clock, d.h. $E(\max(LC(e_i^j) - t(e_i^j), 0)$	$[\mu s]$
$\overline{\mathcal{N}_{C}} = \overline{\overline{C - T}^{-}}$	durchschnittliches Nachgehen der Prozessor-Clock, d.h. $E(\min(C(t(e_i^j)) - t(e_i^j), 0)$	$[\mu s]$
$\frac{\overline{\mathcal{N}_{LC'}}}{\overline{\mathcal{N}_{LC}}} = \frac{\overline{\overline{LC'-T}^{-}}}{\overline{\overline{LC-T}^{-}}}$	Nachgehen der geregelten im Vergleich zur einfachen logical Clock; aufgrund der Definition der geregelten logical Clock muß diese in diesem Parameter besser sein als die einfache logical Clock, d.h. der Quotient ist kleiner 1. **Kriterium: möglichst $\ll$ 1**	[1]

Observable	Bedeutung und Kriterium	Einheit
$\overline{\Uparrow_{LC'}} = \overline{max(LC' - T)}$	Maximales Vorgehen der Uhr LC' in jedem Prozeß – Durchschnitt über alle Prozesse.	$[\mu s]$
$\overline{\Downarrow_{LC'}} = \overline{min(LC' - T)}$	wie $\overline{\Uparrow_{LC'}}$ nur größtes Nachgehen.	$[\mu s]$
$\overline{\Uparrow_{LC}}$, $\overline{\Uparrow_{C}}$, $\overline{\Downarrow_{LC}}$, $\overline{\Downarrow_{C}}$	wie oben, nur mit LC, bzw. C.	$[\mu s]$
$\overline{\mathcal{A}_{LC'}} = \overline{\frac{\sum \lvert \triangle LC' - \triangle T \rvert}{\tau_{ges}}}$	durchschnittliche absolute Abweichung zwischen LC' und T bei der Messung der Zeitintervalle zwischen aufeinanderfolgenden Ereignissen im gleichen Prozeß, d.h. $E(\sum_{j=1,2...} \lvert (LC'_i(e_i^j) - LC'_i(e_i^{j-1})) - (t(e_i^j) - t(e_i^{j-1}))\rvert / \tau_{ges},\ i = 1..n)$ Dies ist ein Maß für die Brauchbarkeit von LC' für beliebige Performancemessungen. **Kriterium: möglichst kleiner 5 %** $= 50/1000$	[1/1000]

Observable	Bedeutung und Kriterium	Einheit
$\overline{\mathcal{A}_{LC}} = \overline{\frac{\sum \lvert\triangle LC - \triangle T\rvert}{\tau_{ges}}}$	wie $\overline{\frac{\sum \lvert\triangle LC' - \triangle T\rvert}{\tau_{ges}}}$ nur mit LC_i statt mit LC'_i.	[1/1000]
$\frac{\overline{\mathcal{A}_{LC'}}}{\overline{\mathcal{A}_{LC}}} = \frac{\overline{\sum \lvert\triangle LC' - \triangle T\rvert}}{\overline{\sum \lvert\triangle LC - \triangle T\rvert}}$	Verhältnis dieser durchschnittlichen absoluten Abweichungen. Ein Wert kleiner 1 bedeutet, daß die geregelte logical Clock besser als die einfache ist. **Kriterium: möglichst $\ll$ 1**	[1]
$\overline{t\mathcal{T}_{LC'}} = \overline{\left(\frac{\sum_{tag=t} \triangle LC' - \triangle T}{\sum_{tag=t} \triangle T}\right)}$	Der Wert gibt den Fehler an, wenn man mittels LC' die Zeitintervalle zwischen den Ereignissen mit tag $= t$ und den direkt davor liegenden Ereignissen im gleichen Prozeß aufsummiert, d.h. $E(\frac{\sum_{j=1,2\ldots\wedge tag(e_i^j)=t}((LC'_i(e_i^j) - LC'_i(e_i^{j-1})) - (t(e_i^j) - t(e_i^{j-1})))}{\tau_{tag=t}},$ $i = 1..n)$ mit $\tau_{tag=t} = \sum_{tag=t} \triangle T =$ $\sum_{j=1,2\ldots\wedge tag(e_i^j)=t}(t(e_i^j) - t(e_i^{j-1}))$ Der Wert gibt also den Fehler bei beispielhaften Performancemessungen an. **Kriterium: möglichst kleiner 5 %** $= 50/1000$	[1/1000]
$\overline{t\mathcal{T}_{LC}} = \overline{\left(\frac{\sum_{tag=t} \triangle LC - \triangle T}{\sum_{tag=t} \triangle T}\right)}$	wie $\overline{\left(\frac{\sum_{tag=t} \triangle LC' - \triangle T}{\sum_{tag=t} \triangle T}\right)}$, nur mit LC_i statt mit LC'_i.	[1/1000]
$\frac{\overline{t\mathcal{T}_{LC'}}}{\overline{t\mathcal{T}_{LC}}} = \frac{\overline{\left(\frac{\sum_{tag=t} \triangle LC' - \triangle T}{\sum_{tag=t} \triangle T}\right)}}{\overline{\left(\frac{\sum_{tag=t} \triangle LC - \triangle T}{\sum_{tag=t} \triangle T}\right)}}$	Verhältnis dieser Fehler. Ein Wert kleiner 1 bedeutet, daß die geregelte logical Clock besser als die einfache ist. **Kriterium: möglichst $\ll$ 1**	[1]
$\overline{\gamma}$	Durchschnittliches $\gamma_{i,j}$, $E(\gamma_{i,j})$. d.h. Der Wert gibt an, wie stark die geregelten logical Clocks durchschnittlich beim *Zurückstellen* verlangsamt werden müssen. $\overline{\gamma} = 95\% \;\hat{=}$ Verlangsamung um nur 5 % $\overline{\gamma} = 90\% \;\hat{=}$ Verlangsamung um nur 10 % $\overline{\gamma} = \;\;0\% \;\hat{=}$ Anhalten der Logical Clock. Der Wert kann niemals über γ_{max} liegen. **Kriterium: möglichst $\overline{\gamma} \geq$ 90%** $= 900/1000$	[1/1000]

Soweit nicht anders spezifiziert, bedeutet

$$E(\text{\_term\_}) = E(E(\text{\_term\_}, j = 0..j_{\max}(i)), i = 1..n)$$

und

$$E(\text{\_term\_}, \text{\_loop\_}) = \left(\sum_{\text{\_loop\_}} \text{\_term\_}\right) / (\text{Länge der \_loop\_})$$

Zu sämtlichen Durchschnittswerten $E(..., i = 1, n)$ wird jeweils auch die Varianz $\sigma(..., i = 1, n)$ ermittelt. Die Varianz ist ein Maß für die Schwankungen der Observablen bei den einzelnen Prozessen. Die Meßwerte werden daher in folgender Form dargestellt.[2]

$$E(...) \pm \sigma(...)$$

6.5 Analyseergebnisse

Es wurden folgende Experimente durchgeführt:

- Variation von τ_{ges} mit Uhrenfehler

  Beim 8. Prozeß von 20 Prozessen geht die Prozessor-Uhr C konstant 1 ms vor.[3] Die Dauer des Experiments τ_{ges} wurde variiert (0.1, 1, 10, 100, 500 s).

  Ergebnis: Die Werte sämtlicher Analyseparameter sind von der Dauer des Experiments unabhängig.

  Bewertung: Einschwingvorgänge des Reglers haben auf die Meßergebnisse bei einem $\tau_{ges} \geq 0.1$s keine Auswirkung.

- Das Standardexperiment mit einer vorgehenden Uhr und $\tau_{ges} = 10$s

  Prozeß 8 geht konstant um 1 ms vor. Tabelle 6.1 auf S. 28 zeigt die Meßwerte für jeden Prozeß und Minimum, Durchschnitt $\oslash$, Varianz σ und Maximum über alle Prozesse. Abschließend sind noch die Quotienten der LC'-Durchschnitte zu den LC-Durchschnitten und die Quotienten der σ-Werte angegeben.

  In kleinerer Schrift sind außerdem unter den Durchschnittszeilen die in Kap. 6.4 auf S. 24 aufgestellten Kriterien angeführt.

  Die Werte von $\overline{LC' - T}^-$, $\overline{LC - T}^-$ und $\overline{C - T}^-$ sind bei allen Prozessen 0.

  Ergebnis:

  Während die geregelte logical Clock um den Faktor 1.9 stärker als die einfache logical Clock vorgeht, ist ihre durchschnittliche absolute Abweichung um den Faktor $3.0 = \frac{101}{34}$ besser. Ebenso sind die Meßwerte für die Zeitintervalle zu den Events mit $tag = t$ um die Faktoren $4.7 = \frac{-99}{-21}$, $4.5 = \frac{-85}{-19}$, $1.9 = \frac{-33}{-17}$ und $2.2 = \frac{65}{29}$ besser.

  Zusätzlich zeigt ein Blick auf die Ergebnisse der einzelnen Prozesse, daß die geregelte logical Clock eine wesentlich gleichförmigere Verteilung der Fehler auf die einzelnen Prozesse realisiert. Dadurch werden die maximalen Meßfehler in den einzelenen Prozessen für die Zeitintervalle zu den Events im Extremfall von $\frac{910}{1000}$ auf $\frac{108}{1000}$ reduziert. Dies sieht man auch deutlich an den bis zum Faktor $9.4 = \frac{254}{27}$ verbesserten Varianzen.

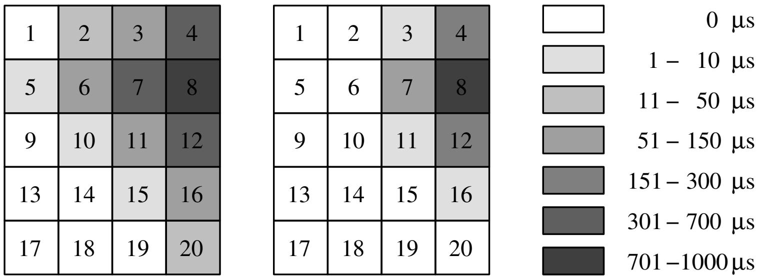
---

[2] Diese Darstellung bedeutet aber nicht, daß die Messwerte in dem Intervall $[E - \sigma, E + \sigma]$ liegen oder dieses ***ausschöpfen***; Bsp.: Eine Meßreihe für $\overline{\gamma}$ ergab für die einzelnen Prozesse die Werte 950, 939, 792, 934, 946, 880, 876, 950, 950, 949, 837, 893, 950, 950, 948, 792, 950, 950, 950, 937, d.h. die Werte liegen zwischen 792 und 950; das zugehörige Meßergebnis ist $\overline{\gamma} = 916 \pm 52$.

[3] Dieser Uhrenfehler kann zwar in Wirklichkeit nicht vorkommen, da er durch den Algorithmus 1 auf S. 9 vollständig ausgeglichen werden würde, aber er stellt einen guten Referenzfehler für die Beurteilung der geregelten logical Clock dar.
Außerdem wurde in allen Experimenten statt 1 ms nur $\Delta = 999\mu$s gewählt, damit in den meisten Tabellen eine 3-stellige Ausgabe genügt.

Proc. No.	$\mathcal{V}_{LC'}$		$\mathcal{V}_{LC}$		$\mathcal{V}_{C}$		$\mathcal{A}_{LC'}$		$\left({}_t\mathcal{T}_{LC'}\ {}_t\mathcal{T}_{LC'}\ \frac{\tau_{tag=t}}{\tau_{ges}}\right)$												$\overline{\gamma}$
		$\Uparrow_{LC'}$		$\Uparrow_{LC}$		$\Uparrow_{C}$		$\mathcal{A}_{LC}$	$tag=1$			$tag=2$			$tag=3$			$tag=4$			
	μs	μs	μs	μs	μs	μs	$\frac{1}{1000}$	$\frac{1}{1000}$	$\frac{1}{1000}$	$\frac{1}{1000}$	$\frac{1}{1000}$	$\frac{1}{1000}$	$\frac{1}{1000}$	$\frac{1}{1000}$	$\frac{1}{1000}$	$\frac{1}{1000}$	$\frac{1}{1000}$	$\frac{1}{1000}$	$\frac{1}{1000}$	$\frac{1}{1000}$	$\frac{1}{1000}$
1	0	0	0	0	0	0	0	0	0	0	26	0	0	87	0	0	524	0	0	363	950
2	15	390	0	0	0	0	14	0	-8	0	25	-7	0	130	-6	0	504	12	0	341	950
3	127	736	1	328	0	0	75	21	-54	0	24	-44	0	130	-30	0	493	62	0	353	942
4	605	905	297	905	0	0	111	824	-70	-910	25	-66	-879	87	-61	-400	459	83	660	429	944
5	2	127	0	0	0	0	4	0	-5	0	25	-4	0	130	-2	0	466	4	0	378	950
6	57	372	0	38	0	0	47	1	-33	0	24	-28	0	174	-22	0	474	49	0	327	950
7	336	951	60	951	0	0	125	401	-66	-317	25	-66	-196	174	-60	-27	436	108	147	365	940
8	999	999	999	999	999	999	0	0	0	0	25	0	0	130	0	0	493	0	0	352	950
9	0	0	0	0	0	0	0	0	0	0	24	0	0	130	0	0	485	0	0	360	950
10	1	93	0	0	0	0	4	0	-2	0	26	-2	0	174	-2	0	433	4	0	367	950
11	95	639	1	244	0	0	69	15	-46	0	24	-37	0	174	-27	0	460	59	0	342	948
12	536	840	197	840	0	0	132	745	-70	-748	25	-70	-632	130	-74	-233	436	106	498	408	938
13	0	0	0	0	0	0	0	0	0	0	27	0	0	130	0	0	458	0	0	385	950
14	0	0	0	0	0	0	0	0	0	0	25	0	0	174	0	0	514	0	0	288	950
15	2	130	0	0	0	0	5	0	-2	0	25	-2	0	174	-2	0	469	4	0	332	950
16	146	625	1	296	0	0	82	18	-53	-3	27	-47	0	130	-41	0	477	74	0	365	944
17	0	0	0	0	0	0	0	0	0	0	25	0	0	87	0	0	478	0	0	410	950
18	0	0	0	0	0	0	0	0	0	0	24	0	0	130	0	0	481	0	0	364	950
19	0	0	0	0	0	0	0	0	0	0	24	0	0	130	0	0	485	0	0	360	950
20	15	247	0	0	0	0	19	0	-15	0	27	-12	0	87	-8	0	463	13	0	423	950
min	0	0	0	0	0	0	0	0	-70	-910	24	-70	-879	87	-74	-400	433	0	0	288	938
⊘	147	352	78	230	50	49	34	101	-21	-99	25	-19	-85	135	-17	-33	474	29	65	366	948
Krit.							<50		<50			<50			<50			<50			>900
σ	263	366	224	362	218	217	46	244	27	254	1	25	231	30	24	98	24	38	176	33	4
max	999	999	999	999	999	999	132	824	0	0	27	0	0	174	0	0	524	108	660	429	950
$\frac{\oslash_{LC'}}{\oslash_{LC}}$	1.9						0.34		0.21			0.22			0.52			0.45			
Krit.	<2						$\ll 1$		$\ll 1$			$\ll 1$			$\ll 1$			$\ll 1$			
$\frac{\sigma_{LC'}}{\sigma_{LC}}$	1.2						0.19		0.11			0.11			0.24			0.22			

Tabelle 6.1: Simulationsergebnisse mit einer vorgehenden Uhr



geregelte Logical Clock Ø=147µs σ=263µs — einfache Logical Clock Ø=78µs σ=224µs

Abb. 6.1: Durchschnittliches Vorgehen $\mathcal{V}_{LC'}$ und $\mathcal{V}_{LC}$ bei

Abb. 6.1 zeigt das durchschnittliche Vorgehen der geregelten und einfachen logical Clock bei allen Prozessen.

Proc. No.	$\mathcal{N}_{LC'}$	$\Downarrow_{LC'}$	$\mathcal{N}_{LC}$	$\Downarrow_{LC}$	$\mathcal{N}_{C}$	$\Downarrow_{C}$	$\mathcal{A}_{LC'}$	$\mathcal{A}_{LC}$	$\left({}_t\mathcal{T}_{LC'}\ {}_t\mathcal{T}_{LC'}\ \frac{\tau_{tag=t}}{\tau_{ges}}\right)$ $tag=1$			$tag=2$			$tag=3$			$tag=4$			$\overline{\gamma}$
	μs	μs	μs	μs	μs	μs	$\frac{1}{1000}$	$\frac{1}{1000}$	$\frac{1}{1000}$	$\frac{1}{1000}$	$\frac{1}{1000}$	$\frac{1}{1000}$	$\frac{1}{1000}$	$\frac{1}{1000}$	$\frac{1}{1000}$	$\frac{1}{1000}$	$\frac{1}{1000}$	$\frac{1}{1000}$	$\frac{1}{1000}$	$\frac{1}{1000}$	$\frac{1}{1000}$
1–7 9–20	0	0	0	0	0	0	0	0	0	0	24–26	0	0	87–174	0	0	433–524	0	0	327–429	950
8	-355	-999	-619	-999	-999	-999	132	961	-78	-1000	25	-71	-987	130	-72	-479	493	133	1108	352	950
$\oslash$	-18	-50	-31	-50	-50	-50	7	48	-4	-50	25	-4	-50	135	-4	-24	474	7	55	366	949
Krit.							<50		<50			<50			<50			<50			>900
$\frac{\oslash_{LC'}}{\oslash_{LC}}$	0.57						0.14		0.08			0.07			0.15			0.12			
Krit.	$\ll$1						$\ll$1		$\ll$1			$\ll$1			$\ll$1			$\ll$1			

Tabelle 6.2: Simulationsergebnisse mit einer nachgehenden Uhr

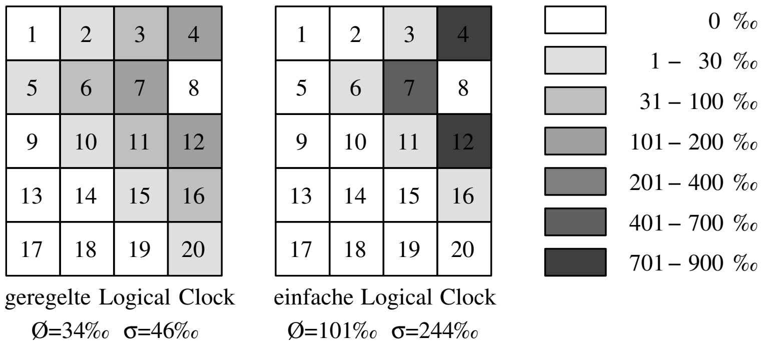


Abb. 6.2: Durchschnittliche absolute Abweichung $\mathcal{A}_{LC'}$ und $\mathcal{A}_{LC}$ bei

Abb. 6.2 zeigt die durchschnittliche absolute Abweichung der geregelten und einfachen logical Clock bei allen Prozessen.

Bewertung:

Die geregelte logical Clock verteilt die Fehler bei einer vorgehenden Uhr sehr gut auf die Nachbarprozesse.

- Das Standardexperiment mit einer nachgehenden Uhr und $\tau_{ges} = 10$s

Prozeß 8 geht konstant um 1 ms nach. Tabelle 6.2 zeigt die Meßwerte für jeden Prozeß und den Durchschnitt $\oslash$ über alle Prozesse, sowie die Quotienten der LC'-Durchschnitte zu den LC-Durchschnitten.

Die Werte von $\overline{LC' - T}^{+}$, $\overline{LC - T}^{+}$ und $\overline{C - T}^{+}$ sind bei allen Prozessen 0.

Ergebnis:

Die einfache und die geregelte logical Clock ändern – wie erwartet – nur die nachgehende Uhr selbst. Die anderen Prozesse sind nicht betroffen.

Die geregelte logical Clock gleicht zu ca. 65 % das Nachgehen der Uhr aus. Die absolute Abweichung wird von 96.1 % auf 13.2 % reduziert. Die Meßfehler für die einzelnen Zeitintervalle wurden auf 7.1 % bis 13.3 % reduziert.

Bei der Durchschnittsbildung über alle Prozesse werden daher die Kriterien weit unterschritten.

Bewertung:

Die geregelte logical Clock reduziert die Fehler einer nachgehenden Uhr sehr stark.

- Variation von τ_{ges} mit Uhrenfehler

  Einer von 20 Prozessoren hat einen Uhrenfehler vom Typ . Die Dauer des Experiments τ_{ges} wurde variiert (0.1, 1, 10, 100, 500 s).

  Ergebnis: Die Ergebnisse für τ_{ges} = 10, 100 und 500 Sekunden sind fast identisch. Bei 1 Sekunde gibt es in $\overline{LC' - T}^+$ Unterschiede von 1 bis 2 μs bei einigen Prozessen und $\overline{\max(LC' - T)}$ ist um ca. 30 μs kleiner als bei τ_{ges} = 10, 100, 500. Bei $\tau_{ges} = 0.1$s ist $\overline{LC' - T}^+$ um 6 μs und $\overline{\max(LC' - T)}$ um 150 μs, bzw. 40% kleiner.

  Bewertung: Das Ergebnis ist bei vorgegebenem Δ bei Drifts kleiner als 2/1000 von der Größe des Drifts unabhängig. Daher ist es möglich, die folgenden Untersuchungen grundsätzlich mit $\tau_{ges} = 10$s durchzuführen.

- Versuche mit den verschiedenen Uhrenfehlern.

  Ergebnis: Die Werte bei , , und sind untereinander nahezu identisch. Ebenso , , und . Und ebenso , , und . Die Ergebnisse bei den letzteren Messungen entsprechen hierbei etwa dem Durchschnitt aus den Messungen der Typen und .

  Dies war nach den Ergebnissen der vorangegangenen Messungen (mit unterschiedlichem Drift) zu erwarten.

  Nur bei den maximalen Abweichungen wirkt sich der +2ms Sprung in aus, d.h. der maximale Fehler in 2 Prozessen liegt bei 1065, bzw. 1227 μs anstatt nur bei 905, bzw. 1000 μs.

  Ein Blick auf die Ergebnisse für die einzelnen Prozessoren zeigt, daß sich erwartungsgemäß nachgehende Uhren nur auf Messungen im zugehörigen Prozeß auswirken, während vorgehende Uhren über den Mechanismus der einfachen logical Clock sich auf die nächsten und übernächsten Nachbarn auswirken. Bei der geregelten logical Clock sind außerdem noch die überübernächsten Nachbarn in diesen Beispielen betroffen.

  Ein Vergleich von mit zeigt, daß Zeiten mit größerem Fehler bei C sich überproportional auf den Fehler von LC' auswirken und damit ein schlechteres Resultat für die Bewertung der geregelten logical Clock ergibt.

  Bewertung: Für die weiteren Experimente wird das Standardexperiment mit Uhrenfehler als Vergleichsbasis eingesetzt.

- Variation von $ex\frac{1}{2}\_time\_^{min}_{max}$ und Δ

  Der 8. Prozeß von 20 hat den Uhrenfehler . Es werden die Parameter $ex\frac{1}{2}\_time\_^{min}_{max}$ und Δ variiert. Für die Bewertung wurden die Zeiten zu $tag = 1$ herangezogen, da hierbei das vorangehende Receive-Event die Meßfehler am stärksten beeinflußt.

  Ergebnis: Tabelle 6.3 auf S. 31 zeigt die Meßwerte im Detail.

  Es zeigt sich, daß die Meßfehler bei einer Vergrößerung der Executiontime Parameter abnehmen. Für größere Werte von Δ wurden die Executiontime Parameter

$\frac{ex_2^1\_time\_^{min}_{max}}{Default}$	[1]	0.5	1	2	5	5	10	10	20	50	100	Krit.
Δ	[ms]	1	1	1	1	2.75	1	4.7	8.5	15	27	
$\overline{\mathcal{V}_{LC'}}$	[μs]	213 ±282	147 ±263	100 ±236	78 ±221	321 ±663	69 ±218	543 ±1128	970 ±2029	1551 ±3494	2672 ±6238	
$\overline{\mathcal{V}_{LC}}$	[μs]	91 ±231	78 ±224	64 ±218	59 ±217	220 ±608	57 ±217	383 ±1040	693 ±1880	1142 ±3297	2021 ±5928	
$\overline{\mathcal{V}_{C}}$	[μs]	50 ±218	50 ±218	50 ±218	50 ±218	137 ±599	50 ±218	235 ±1024	425 ±1853	750 ±3269	1350 ±5885	
$\overline{\mathcal{V}_{LC'}}/\overline{\mathcal{V}_{LC}}$	[1]	2.3 ±1.2	1.9 ±1.2	1.6 ±1.1	1.3 ±1.0	1.5 ±1.1	1.2 ±1.0	1.4 ±1.1	1.4 ±1.1	1.4 ±1.1	1.3 ±1.1	< 2
$\overline{\mathcal{A}_{LC'}}$	[$\frac{1}{1000}$]	65 ±62	34 ±46	15 ±27	7 ±15	37 ±56	4 ±9	35 ±53	33 ±51	24 ±39	21 ±38	$< \frac{50}{1000}$
$\overline{\mathcal{A}_{LC}}$	[$\frac{1}{1000}$]	166 ±363	101 ±244	36 ±91	13 ±31	74 ±156	6 ±14	67 ±138	63 ±128	40 ±85	35 ±76	
$\overline{\mathcal{A}_{LC'}}/\overline{\mathcal{A}_{LC}}$	[1]	0.39 ±0.17	0.33 ±0.19	0.42 ±0.23	0.53 ±0.48	0.50 ±0.36	0.7 ±0.6	0.52 ±0.38	0.52 ±0.40	0.60 ±0.46	0.60 ±0.50	$\ll 1$
$\overline{{}_1\mathcal{T}_{LC'}}$	[$\frac{1}{1000}$]	-48 ±43	-21 ±27	-9 ±15	-4 ±8	-21 ±31	-2 ±5	-21 ±33	-21 ±35	-21 ±34	-21 ±38	$< \frac{50}{1000}$
$\overline{{}_1\mathcal{T}_{LC}}$	[$\frac{1}{1000}$]	-133 ±310	-99 ±254	-46 ±123	-32 ±79	-71 ±146	-23 ±59	-71 ±146	-68 ±138	-53 ±109	-51 ±106	
$\overline{{}_1\mathcal{T}_{LC'}}/\overline{{}_1\mathcal{T}_{LC}}$	[1]	0.36 ±0.14	0.21 ±0.11	0.20 ±0.12	0.13 ±0.10	0.30 ±0.21	0.09 ±0.08	0.30 ±0.23	0.31 ±0.25	0.40 ±0.31	0.41 ±0.36	$\ll 1$
$\overline{\gamma}$	[$\frac{1}{1000}$]	943 ±12	948 ±4	950 ±1	950 ±0	925 ±33	950 ±0	914 ±44	902 ±58	897 ±68	890 ±76	$> \frac{900}{1000}$

Tabelle 6.3: Variation von $ex_2^1\_time\_^{min}_{max}$ und Δ.

derart vergrößert, daß die Fehler bei $tag = 1$ wieder in etwa den Werten in der 2. Meßwertespalte entsprechen und die geforderten Kritierien erfüllen.

Bewertung: Der Algorithmus kann nur sinnvoll eingesetzt werden, wenn die Uhrenfehler Δ in einem vernünftigen Verhältnis zu den zu messenden Zeitintervallen bleiben.

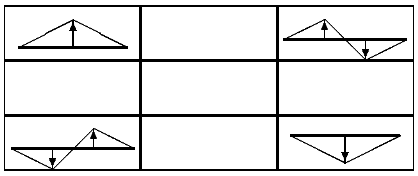
zu messende Zeitintervalle im Bereich								
	von	0.01	0.05	0.1	0.2	0.5	1	ms
	bis	1	5	10	20	50	100	ms
zulässige Uhrenabweichung		1	2.75	4.7	8.5	15	27	ms

Bei größeren durch Driftschwankungen hervorgerufenen Uhrenfehlern[4] ist daher der Einsatz einer Uhrensynchronisation notwendig. Hierbei genügen Methoden, wie z.B. xntp [Mil92], deren Fehler im Bereich von 1 ms liegen und die den Drift zum Zwecke der Synchronisation stark ändern.

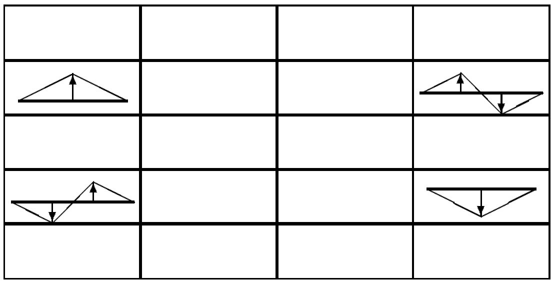
[4]Fehler durch einen konstanten Drift werden durch Algorithmus 1 auf S. 9 vollständig ausgeglichen

- Mehrere fehlerbehaftete Uhren

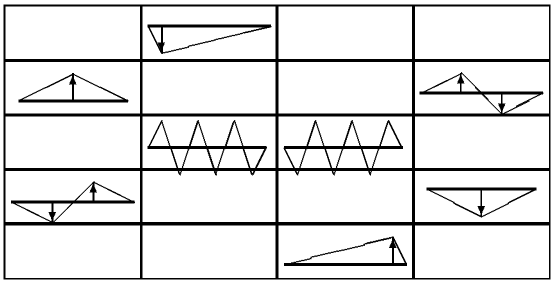
  Hierzu wurden Experimente mit 3×3, 4×5 und 6×7 Prozessen durchgeführt. Die Diagramme geben die Position und Art der falschgehenden Uhren an.



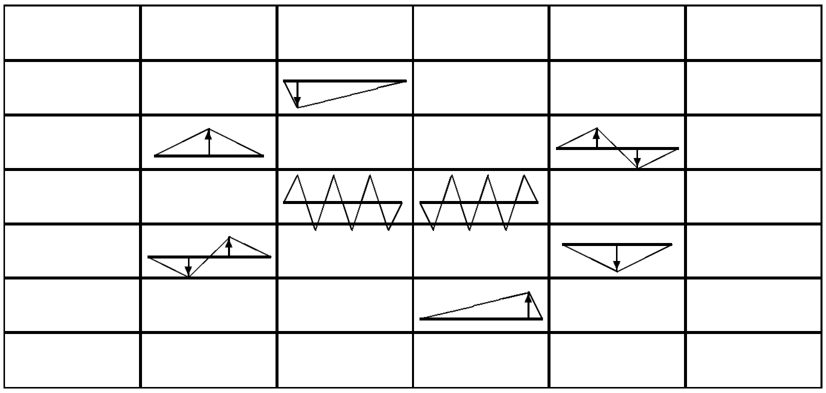
Experiment 1



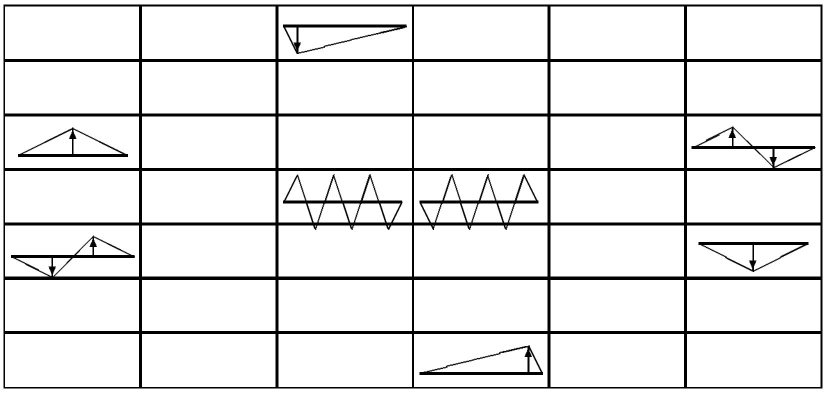
Experiment 2



Experiment 3



Experiment 4



Experiment 5

Ergebnis: Tabelle 6.4 auf S. 33 zeigt die Meßwerte im Detail. Die Kriterien wurden eingehalten oder nur unwesentlich überschritten.

Experiment		1	2	3	4	5	Krit.
$\overline{\mathcal{V}_{LC'}}$	[μs]	179 $\pm$149	89 $\pm$127	207 $\pm$132	123 $\pm$122	103 $\pm$121	
$\overline{\mathcal{V}_{LC}}$	[μs]	127 $\pm$169	61 $\pm$124	127 $\pm$153	63 $\pm$120	57 $\pm$121	
$\overline{\mathcal{V}_{LC'}}/\overline{\mathcal{V}_{LC}}$	[1]	1.4 $\pm$0.88	1.6 $\pm$1.0	1.6 $\pm$0.86	1.95 $\pm$1.02	1.8 $\pm$1.0	< 2
$\overline{\mathcal{N}_{LC'}}$	[μs]	-40 $\pm$61	-23 $\pm$61	-29 $\pm$53	-16 $\pm$46	-17 $\pm$49	
$\overline{\mathcal{N}_{LC}}$	[μs]	-87 $\pm$133	-40 $\pm$103	-71 $\pm$126	-35 $\pm$98	-33 $\pm$92	
$\overline{\mathcal{N}_{LC'}}/\overline{\mathcal{N}_{LC}}$	[1]	0.46 $\pm$0.46	0.58 $\pm$0.59	0.41 $\pm$0.42	0.46 $\pm$0.47	0.52 $\pm$0.53	$\ll 1$
$\overline{\mathcal{A}_{LC'}}$	[$\frac{1}{1000}$]	55 $\pm$43	30 $\pm$29	68 $\pm$42	47 $\pm$39	35 $\pm$27	$< \frac{50}{1000}$
$\overline{\mathcal{A}_{LC}}$	[$\frac{1}{1000}$]	153 $\pm$122	85 $\pm$102	198 $\pm$120	108 $\pm$118	91 $\pm$114	
$\overline{\mathcal{A}_{LC'}}/\overline{\mathcal{A}_{LC}}$	[1]	0.36 $\pm$0.35	0.35 $\pm$0.28	0.34 $\pm$0.35	0.44 $\pm$0.33	0.38 $\pm$0.24	$\ll 1$

Tabelle 6.4: Mehrere fehlerbehaftete Uhren.

- Variation der Reglerparameter

  Die Tabelle 6.5 auf S. 34 zeigt, daß die Standardwerte eine ausgewogene Einstellung darstellen, und daß Änderungen dieser Werte das Ergebnis der geregelten logical Clock nur geringfügig beeinflussen.

  Der Bereich der Variation jedes Reglerparameters[5] ist in etwa vergleichbar mit der Variation des Uhrenfehlerparameters Δ zwischen 0.8 und 1.0 ms. Die Änderungen der Reglerparameter wirken sich im Vergleich zur Änderung von Δ nur sehr gering aus.

  Bewertung: Es ist wahrscheinlich, daß die gewählten Reglerparameter für die meisten Uhrenfehler und Anwendungen eine gute Einstellung darstellen.

- Nichtkontinuierliche Uhren

  Bei den Experimenten wurde eine Tickdauer von 1 ms zugrunde gelegt. Die einzelnen Uhren waren versetzt zueinander.

  Ergebnis: Bei 4 nichtkontinuierlichen Uhren von 20 liegt das Vor- bzw. Nachgehen bei $\overline{\mathcal{V}_{LC'}}$ =44$_{\pm 57}$ bzw. $\overline{\mathcal{N}_{LC'}}$ =-20$_{\pm 41}$ μs und die Meßfehler bei den einzelnen Tags bei $\overline{_t\mathcal{T}_{LC'}}$ =1.2$_{\pm 1.3}$ %. Wenn alle 20 Uhren nichtkontinuierlich sind, dann erhöhen sich die Uhrenfehler auf 298$_{\pm 22}$ bzw. -39$_{\pm 6}$ μs und bei den Tags auf 13$_{\pm 3}$ %. Durch die Regelung der Logical Clock war kein Gewinn erzielbar.

  Ein Vergleich von $\overline{_t\mathcal{T}_{LC'}}$ mit $\overline{_t\mathcal{T}_C}$ zeigt, daß trotz des Uhrenablesefehlers von $\pm 500\,\mu$s die direkt mittels C gewonnenen Ergebnisse um den Faktor 30 und $\overline{|_t\mathcal{T}_C|}$ um den Faktor 7 besser sind.[6]

[5] Hierzu sind bei γ_{max}, $\gamma_{degress}$ und q_{factor} die Variationen der Differenzen zu 1 zu vergleichen

[6] Bei allen anderen Uhrenfehlern sind die Fehler $\overline{_t\mathcal{T}_C}$ und $\overline{|_t\mathcal{T}_C|}$ nahezu 0.

Reglerparameter	geänderter Wert	Standard-Wert	$\overline{\overline{LC' - T}^{+}}$	$\overline{\left(\frac{\sum_{tag=1} \Delta LC' - \Delta T}{\sum_{tag=1} \Delta T}\right)}$	Bewertung
Standard			147 ± 263	-21 ± 27	
γ_{max}	0.93	0.95	137 ± 259	-23 ± 32	$+-$
γ_{max}	0.94	0.95	142 ± 261	-22 ± 29	$+-$
γ_{max}	0.96	0.95	158 ± 266	-20 ± 24	$-+$
γ_{max}	0.97	0.95	165 ± 267	-21 ± 24	$-o$
$\gamma_{degress}$	0.80	0.90	145 ± 262	-21 ± 27	$+o$
$\gamma_{degress}$	0.85	0.90	147 ± 263	-21 ± 27	oo
$\gamma_{degress}$	0.95	0.90	149 ± 264	-21 ± 27	$-o$
$\gamma_{degress}$	0.97	0.90	149 ± 264	-21 ± 27	$-o$
$\gamma_{degress}$	0.99	0.90	152 ± 265	-22 ± 26	$--$
l_{upper}	1.80	2.00	142 ± 260	-22 ± 29	$+-$
l_{upper}	2.20	2.00	153 ± 265	-22 ± 27	$--$
l_{lower}	1.50	1.80	145 ± 263	-22 ± 28	$+-$
l_{lower}	2.00	1.80	148 ± 263	-21 ± 26	$-o$
q_{init} u. q_{min}	200	250	140 ± 261	-24 ± 32	$+-$
q_{init} u. q_{min}	300	250	155 ± 265	-21 ± 26	$-o$
q_{init} u. q_{min}	400	250	163 ± 266	-22 ± 25	$--$
q_{init} u. q_{min}	500	250	163 ± 266	-22 ± 25	$--$
q_{init}	350	250	147 ± 263	-21 ± 27	oo
q_{init}	500	250	147 ± 263	-21 ± 27	oo
q_{factor}	0.85	0.90	142 ± 260	-22 ± 29	$+-$
q_{factor}	0.88	0.90	145 ± 262	-22 ± 28	$+-$
q_{factor}	0.92	0.90	148 ± 265	-21 ± 27	$-o$
q_{factor}	0.95	0.90	152 ± 267	-22 ± 27	$--$
Im Vergleich hierzu die Wirkung variierter Δ-Werte					
Δ	0.4	1 ms	27 ± 87	-4 ± 10	
Δ	0.6	1 ms	56 ± 139	-9 ± 19	
Δ	0.8	1 ms	98 ± 200	-14 ± 22	
Δ	1.0	1 ms	147 ± 263	-21 ± 27	
Δ	1.2	1 ms	206 ± 328	-34 ± 40	
Δ	1.5	1 ms	312 ± 425	-65 ± 72	

Tabelle 6.5: Variation der Reglerparameter.

Bewertung: Bei nichtkontinuierlichen Uhren kann die geregelte logical Clock eingesetzt werden. Sie erzielt aber gegenüber der einfachen logical Clock keinen Gewinn. Außerdem hat sie keinen Nutzen beim Ausgleich der Uhrenablesefehlers der nichtkontinuierlichen Uhren.

7 Anwendung der geregelten logical Clock

Die geregelte logical Clock ist für Monitoring und Debugging Tools konzipiert.

Beim *Monitoring* verteilter und paralleler Anwendungen kann sie als Filter eingesetzt werden. Ein Tracefile mit Rückwärtsbezügen, d.h. dessen Zeitstempel die Clock Condition (3.4) nicht erfüllen, wird nachträglich so modifiziert, daß er einerseits die Clock Condition erfüllt, und daß andererseits die Zeitstempel für Zeitmessungen nur unwesentlich verändert werden.

Da die geregelte logical Clock zuerst nur für eine Bearbeitung der Events vom Beginn an entworfen ist, gibt es beim Einsatz innerhalb eines Monitortools Probleme, wenn eine beliebige Positionierung innerhalb des Tracefiles mit anschließendem Vorwärts- oder Rückwärtsscrollen möglich sein soll. Hierzu kann man die geregelte logical Clock, bestehend aus den Algorithmen 2, 3 und 4, an einer beliebigen Stelle innerhalb des Traces beginnen lassen. Für das Rückwärtsscrollen kann man den Algorithmus ebenfalls einsetzen, indem man das Vorzeichen der Zeitstempel wechselt und die Rolle von Send- und Receive-Events vertauscht. Dieses Verfahren hat den Nachteil, daß die durch die geregelte logical Clock modifizierten Zeitstempel von der Wahl des Anfangsevents abhängen, d. h. der Anwender sieht das gleiche Event u. U. mit unterschiedlichen Zeitstempeln dargestellt.

Dieser Nachteil kann ausgeglichen werden, indem das Monitortool jedem Ereignis zwei verschiedene Zeitstempel zuordnet,

- die Zeit der lokalen Prozessoruhr, ggf. linear korrigiert mittels Algorithmus 1; sie kann für Zeitdifferenzmessungen innerhalb eines Prozesses und zum Referenzieren eingesetzt werden;
- die Zeit der geregelten logical Clock; sie dient zur Darstellung der Ereignisse in Zeitdiagrammen und für Zeitdifferenzmessungen zwischen Ereignissen in verschiedenen Prozessen; ihr Wert ist jeweils abhängig vom Anfangsevent nach der letzten Repositionierung.

Die geregelte logical Clock kann auch innerhalb von zentralen *Debug-Tools für parallele Programme* eingesetzt werden. Die fehlende Synchronisation am Ende muß hierbei durch eine resourcenarme fortlaufende Synchronisation parallel zur Anwendung ersetzt werden. Die Synchronisation am Anfang kann dann in der Regel ebenfalls entfallen.

Es wird empfohlen, eine Abschätzung für das minimale Delay als Nebenprodukt des eingesetzten Synchronisationsalgorithmus zu erlangen. Die Streuung der Messagedelays besitzt noch einen signifikanten Bereich unterhalb der Delays mit der größten Häufigkeit. Unsere Messungen haben in einem Ethernet und in einem FDDI-Ring ergeben, daß das minimale Delay im allgemeinen größer als ein Viertel der häufigsten Roundtriptime eines leeren RPCs ist.

Weitere Anwendungsbereiche sind dadurch möglich, daß die Algorithmen 2, 3 und 4 so konzipiert sind, daß sie wie Lamports Logical Clock auch mittels einer Protokollergänzung direkt in eine parallelen Anwendung integriert werden können. Hierzu sind nur die schon berechneten Zeitstempel $LC_k(e_k^l)$ und $LC'_k(e_k^l)$ in den Messages $(e_k^l, e_i^j) \in M$ zu übertragen. Damit stehen für die Algorithmen 2 und 3 alle Daten zur Verfügung. Der Regler (Algorithmus 4) kann im Prinzip auch lokal arbeiten. Der Schritt-Index s ist dann durch den Event-Index j zu ersetzen.

8 Zusammenfassung

Die geregelte logical Clock ist eine Methode zum Korrigieren der Zeitstempel mit folgenden Eigenschaften

- Die korrigierten Zeitstempel erfüllen die Clock Condition.

  D.h. ein causal nachfolgendes Ereignis hat auch einen späteren Zeitstempel. Die neuen Zeitstempel sind damit zur Darstellung der Ereignisse in einem Monitortool geeignet.

- Geringes Vorgehen und noch geringeres Nachgehen.

  Damit kann sie problemlos für durchschnittliche Zeitmessungen zwischen Ereignissen zweier verschiedner Prozesse eingesetzt werden, z.B. zur Messung der Dauer eines Nachrichtentransfers.

- Geringe Meßfehler für durchschnittliche Zeitspannen zwischen Ereignissen im selben Prozeß.

  Damit entfällt die Notwendigkeit, für die Darstellung der Ereignisse, für die Zeitmessungen zwischen verschiedenen Prozessen und für die Zeitmessungen innerhalb der Prozesse unterschiedliche Zeitstempel zu benutzen.

Die geregelte logical Clock kann eingesetzt werden, wenn die Uhren mit einer maximalen Differenz von ca. zwei Millisekunde synchronisiert sind. Derartige Synchronisationsmethoden, wie z.B. XNTP, sind resourcenarm. Sie sind meistens zu anderen Zwecken in Workstationclustern schon installiert. Die geregelte logical Clock ist hierbei unempfindlich, wenn zum Zwecke der Synchronisation die Uhrengeschwindigkeit um einige Prozent geändert wird. In Kombination mit solchen Uhrensynchronisationsverfahren kann eine zusätzliche Synchronisation am Anfang und Ende des Programmlaufs entfallen.

Vor allem bei längeren Programmlaufzeiten ist die geregelte logical Clock damit anderen Methoden überlegen, da diese meist eine geringen Varianz der Uhrengeschwindigkeiten voraussetzen.

Typische Einsatzgebiete sind das Monitoring und Debugging paralleler Anwendungen in Workstation-Clustern. Auf Grund ihres geringen zusätzlichen Protokoll- und Rechenaufwands kann sie aber auch in verteilten Anwendungen eingesetzt werden, in denen seither eine diskrete oder teilweise stetige Logical Clock eingesetzt wurde und in denen die Vorzüge der geregelten logical Clock wünschenswert sind.

9 Zukünftige Arbeiten

Es ist geplant, die geregelte logical Clock in der Praxis als Filter für Monitor-Tracefiles zu testen.

Das Prinzip des derzeitigen Reglers begrenzt γ auf einen Wert unter 1, in der Praxis auf 0.95. Dadurch fällt die geregelte Logical Clock immer wieder auf die Prozessor-Clock C_i zurück. Es ist geplant, mit einem geänderten Regler auch $\gamma = 1$ oder zumindest $\gamma = 1 - 10^{-5}$ zuzulassen.

Weitere Fragen ergeben sich beim Einsatz der geregelten logical Clock in Systemen, die *one-sided Communication*, also das Lesen (*get*) und Schreiben (*put*) in dem lokalen Memory eines entfernten Prozesses, gestatten. In [MKSF96] wird hierzu ein ATM-Adapter beschrieben der die put-Operationen größtenteils in Hardware realisiert. Neben der Problematik, daß hierbei im entfernten Prozeß in der Regel kein Trace-Event aufgezeichnet werden kann, muß man auch die Auswirkungen der unterschiedlichen Latencies beim Message Passing und bei der one-sided Communication betrachten.

Literatur

[BD87] Özalp Babaoğlu and Rogério Drummond. (Almost) no cost clock synchronization. In *Proceedings of 7th International Symposium on Fault-Tolerant Computing*, pages 42–47. IEEE Computer Society Press, July 1987.

[BDM+94] P. Baldy, H. Dicky, R. Medina, M. Morvan, and J.F. Vilarem. Efficient reconstruction of the causal relationship in distributed systems. In Michel Cosnard et al., editor, *Parallel and Distributed Computing, Theory and Practice, First Canada-France Conference, Montréal, May 19-21, 1994*, pages 101–113. Springer-Verlag, 1994.

[CB91] Bernardette Charron-Bost. Concerning the size of logical clocks in distributed systems. *Information Processing Letters*, 39(7):11–16, July 1991.

[CF94] Flaviu Cristian and Christof Fetzer. Probabilistic internal clock synchronization. In *Proceedings. 13th Symposium on Reliable Distributed Systems, Dana Point, CA, USA, Oct. 25-27, 1994*, pages 22–31. IEEE Computer Society Press, 1994.

[CF95] Flaviu Cristian and Christof Fetzer. Probabilistic internal clock synchronization. Technical Report CS94-367, University of California, San Diego, May 18 1995. ftp://cs.ucsd.edu/pub/team/internalProbClockSync.ps.Z, ftp://cs.ucsd.edu/pub/cfetzer/CS94-367.ps.Z.

[Che92] Wing-Hong Cheung. Improvements on Fidge logical clocks. In *TENCON '92. Technology Enabling Tomorrow 1992 IEEE Region 10 International Conference. Computers, Communications and Automation towards the 21st Century, Melbourne, Nov. 11-13, 1992*, pages 136–140, vol. I. IEEE, New York, 1992.

[CHK92] Janice E. Cuny, Alfred A. Hough, and Joydip Kundu. Logical time in visualizations produced by parallel programs. In *Proceedings. Visualization '92, Boston, MA, USA, Oct. 19-23, 1992*, pages 186–193. IEEE Computer Society Press, 1992.

[DB93] Rogério Drummond and Özalp Babaoğlu. Low-cost clock synchronization. *Distributed Computing*, 6(4):193–203, July 1993.

[DHHB87] A. Duda, G. Harrus, Y. Haddad, and G. Bernard. Estimating global time in distributed systems. In *Proceedings of the 7th International Conference on Distributed Computing Systems, Berlin, September 21-25, 1987*, pages 299–306. IEEE Computer Society Press, 1987.

[DS93] Curtis E. Dyreson and Richard T. Snodgrass. Timestamp semantics and representation. *Information Systems*, 18(3):143–166, April 1993.

[Dun91] Thomas H. Dunigan. Hypercube clock synchronization. Technical Report ORNL TM-11744, Oak Ridge National Laboratory, TN, February 1991.

[Dun94] Thomas H. Dunigan. Hypercube clock synchronization. ORNL TM-11744 (updated), September 1994. http://www.epm.ornl.gov/~dunigan/clock.ps.

[DW91] G.J.W. van Dijk and A.J. van der Wal. Partial ordering of sychronization events for distributed debugging in tightly-coupled multiprocessor systems. In A. Bode, editor, *Distributed Memory Computing, 2nd European Conference, EDMCC2, Munich, FRG,* LNCS 487, pages 100–109. Springer-Verlag, April 22-24 1991.

[EK94] Dennis Edwards and Phil Kearns. DTVS: a distribute trace visualization system. In *Proceedings. Sixth IEEE Symposium on Parallel and Distributed Processing, Dallas, Oct. 26-29, 1994*, pages 281–288. IEEE Computer Society Press, 1994.

[Fid88] Colin J. Fidge. Timestamps in message-passing systems that preserve partial ordering. In *Proceedings of 11th Australian Computer Science Conference*, pages 56–66, February 1988.

[Fid89] Colin J. Fidge. Partial orders for parallel debugging. *ACM SIGPLAN Notices*, 24(1):183–194, January 1989.

[Hof93a] Richard Hofmann. Gemeinsame Zeitskala für lokale Ereignisspuren. In B. Walke and O. Spaniol, editors, *Messung, Modellierung und Bewertung von Rechen- und Kommunikationssystemen, 7. GI/ITG-Fachtagung, Aachen, 21.-23. September 1993.* Springer-Verlag, Berlin, 1993. ftp://faui79.informatik.uni-erlangen.de/pub/doc/mmb93_globtime.ps.Z.

[Hof93b] Richard Hofmann. Gesicherte Zeitbezüge für die Leistungsanalyse in parallelen und verteilten Systemen. Dissertation, Universität Erlangen-Nürnberg, Technische Fakultät, 1993. ftp://faui79.informatik.uni-erlangen.de/pub/doc/immd26#3.ps.Z.

[HW88] Dieter Haban and Wolfgang Weigel. Global events and global breakpoints in distributed systems. In *Proceedings of 21st Hawaii International Conference on System Sciences*, pages 166–175, vol. II, 1988.

[Jez89a] Jean-Marc Jézéquel. Building a global time on parallel machines. In J.-C. Bermond and M. Raynal, editors, *Proceedings of the 3rd International Workshop on Distributed Algorithms,* LNCS 392, pages 136–147. Springer-Verlag, 1989.

[Jez89b] Jean-Marc Jézéquel. *Outils pour l'expérimentation d'algorithmes distribués sur machines parallèles.* PhD thesis, Université de Rennes, 1. Oct. 1989.

[Lam78] Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 21(7):558–565, July 1978.

[LK93] William S. Lloyd and Phil Kearns. Tracing the execution of distributed programs. *Journal of Systems and Software*, 21(3):201–214, June 1993.

[Mat89] Friedemann Mattern. Virtual time and global states of distributed systems. In M. Cosnard and P. Quinton, editors, *Proceedings of International Workshop on Parallel and Distributed Algorithms, Chateau de Bonas, France, October 1988*, pages 215–226. Elsevier Science Publishers B. V., Amsterdam, 1989.

[Mil92] David L. Mills. Network time protocol (version 3), specification, implementation and analysis. RFC 1305, Request for Comments, March 1992.

[MKSF96] Todd Mummert, Corey Kosak, Peter Steenkiste, and Allan Fisher. Fine grain parallel communication on general purpose LANs. In *International Conference on Supercomputing, ACM, Philadelphia*, May 1996. http://www.cs.cmu.edu/afs/cs/project/iwarp/archive/nectar-papers/96ics.ps.

[MSV91] Sigurd Meldal, Sriram Sankar, and James Vera. Exploiting locality in maintaining potential causality. In *Proceedings of the 10th Annual ACM Symposium on Principles of Distributed Computing*, pages 231–239, 1991. Also Technical Report CSL-TR-91-446, Stanford University, April 1991.

[MT95] Eric Maillet and Cécile Tron. On efficiently implementing global time for performance evaluation on multiprocessor systems. *Journal of Parallel and Distributed Computing*, 28:84–93, 1995.

[PYS92] Robert L. Probert, Hualong Yu, and Kassem Saleh. Relative-clock-based specification and test result analysis of distributed systems. In *Eleventh Annual International Phoenix Conference on Computers and Communications, Scottsdale, AZ, USA, April 1-3, 1992*, pages 687–694. IEEE, New York, 1992.

[Ray87] Michel Raynal. A distributed algorithm to prevent mutual drift between n logical clocks. *Information Processing Letters*, 24:199–202, 1987.

[RS95] Michel Raynal and Mukesh Singhal. Logical time: A way to capture causality in distributed systems. Technical Report OSU-CISRC-4/94-TR13, Ohio State University, Computer and Information Science Research Center, 1995. //ftp://ftp.cis.ohio-state.edu/pub/tech-report/1995/TR13.ps.gz.

[RTV93] Maurice van Riek, Bernard Tourancheau, and Xavier-Francois Vigouroux. Monitoring of distributed memory multicomputer programs (A general approach to the monitoring of distributed memory MIMD multicomputers). Technical Report CS-93-204, University of Tennessee, 1993. //http://www.netlib.org/tennessee/ut-cs-93-204.ps.

[Sch87] Fred B. Schneider. Understanding protocols for byzantine clock synchronization. Technical Report 87-859, Department of Computer Science, Cornell University, August 1987. http://cs-tr.cs.cornell.edu/TR/CORNELLCS:TR87-859/Print.

[SES89] André Schiper, Jorge Eggli, and Alain Sandoz. A new algorithm to implement causal ordering. In J.-C. Bermond and M. Raynal, editors, *Proceedings of the 3rd International Workshop on Distributed Algorithms,* LNCS 392, pages 219–232. Springer-Verlag, 1989.

[SK90] Mukesh Singhal and Ajay Kshemkalyani. An efficient implementation of vector clocks. Technical Research Report OSU-CISRC-11/90-TR34, Computer and Information Science Research Center, Ohio State University, 2036 Neil Avenue Mall, Columbus, Ohio 43210, 1990.

[SM94] Reinhard Schwarz and Friedemann Mattern. Detecting causal relationships in distributed computations: in search of the holy grail. *Distributed Computing*, 7(3):149–174, 1994.

[YM93] Zhonghua Yang and Anthony Marsland, T.˙Annotated bibliography on global states and times in distributed systems. *Operating Systems Review*, 27(3):55–74, July 1993.

[ZEA93] M. Zaki, M.Y. El-Nahas, and H.A. Allam. DPDP: an interactive debugger for parallel and distributed processing. *Journal of Systems and Software*, 22(1):45–61, July 1993.