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 LOGISCHE UHR, EINE GLOBALE UHR FÜR DIE TRACEBASIERTE ÜBERWACHUNG PARALLELER ANWENDUNGEN

Von der Fakultät Informatik der Universität Stuttgart
zur Erlangung der Würde eines Doktors der
Naturwissenschaften (Dr.rer.nat.) genehmigte Abhandlung

vorgelegt von

Rolf Rabenseifner
aus Stuttgart

Hauptberichter:	Prof. Dr. K. Rothermel
Mitberichter:	Prof. Dr.-Ing. R. Rühle
Tag der Einreichung:	30. März 1999
Tag der mündlichen Prüfung:	3. Februar 2000

ISSN 0941-4665 März 2000 RUS - 44

Rechenzentrum der Universität Stuttgart
Allmandring 30
D-70550 Stuttgart

Eine elektronische Version dieser Veröffentlichung mit farbigen Abbildungen und weitere Informationen findet man unter den URLs `http://www.hlrs.de/people/rabenseifner/` und `http://www.hlrs.de/people/rabenseifner/dissertation/`
Die offizielle elektronische Veröffentlichung erhält man unter
`http://elib.uni-stuttgart.de/opus/`

Danksagung

Es ist mir eine Freude, mich bei all den Personen zu bedanken, die meine Arbeit in mannigfacher Weise unterstützt haben. Allen voran gilt mein herzlicher Dank meinen Eltern für ihr Engagement und ihre Liebe und dafür, daß sie in mir die Freude am Entdecken stärkten. Mein besonderer Dank gilt auch meiner Frau Rose Volz-Rabenseifner für ihr Verständnis, mit dem sie meine Arbeit stets unterstützte. Mehr als einmal hat sie das Manuskript mit größter Sorgfalt gelesen und konstruktiv kritisiert. Herzlich danke ich auch unseren beiden Kindern Katja und Jan für ihre Geduld während dieser Zeit.

Meinem Betreuer, Prof. Dr. Andreas Reuter, möchte ich für sein Interesse an meinem Thema und für seine Anregungen, sowie für seine Zuversicht in die Realisierbarkeit dieses Projekts herzlich danken. Prof. Dr. Kurt Rothermel danke ich für die anregende Diskussion und die hilfreichen Hinweise bei der Fertigstellung des Manuskripts, sowie für die Übernahme des Hauptberichts. Für die Übernahme des Mitberichtes und für die Bereitstellung der Ressourcen für diese Arbeit möchte ich mich bei Prof. Dr.-Ing. Roland Rühle bedanken.

Mein Dank gilt ebenfalls Prof. Dr. Wolfgang E. Nagel für die Einladung zu einem Forschungsaufenthalt an der TU Dresden und für die fruchtbare Zusammenarbeit. Bei Prof. Dr. Friedemann Mattern bedanke ich mich für die Einladung zum Instituts-Kolloquium und – auch bei Dr. Reinhard Schwarz – für die Anregungen in der anschließenden Diskussion. Bei Prof. Dr.-Ing. Michael Zeitz danke ich für seinen Schnellkurs in Regelungstechnik. Bei Dr.-Ing. Richard Hofmann bedanke ich mich für sein Interesse an meinen Ideen zur Weiterentwicklung seines Verfahrens. Hans-Christian Hoppe und der Pallas GmbH danke ich für die Bereitstellung des I/O-Interfaces für Vampir-Tracefiles, sowie eines Tracefiles für Testzwecke.

Meinen Kollegen vom Rechenzentrum der Universität Stuttgart danke ich für die gute Zusammenarbeit. Insbesondere danke ich Paul Christ für seine Unterstützung im Rahmen des DFN-RPC Projekts, welches die ursprüngliche Motivation für die Problemstellung dieser Arbeit bildete. Dr. Jörg Hertzer und Walter Wehinger danke ich für die Unterstützung zum Uhrensynchronisationsprogramm xntp.

Maurice van Riek, Prof. Bernard Tourancheau und Xavier-Francois Vigouroux danke ich für ihren hervorragenden Übersichtsbericht zum Monitoring, welcher mich zur Bearbeitung meines Themas anregte. Dr. Yun Ding danke ich für ihre Hilfe bei der Vorbereitung meiner mündlichen Prüfung.

Schließlich möchte ich es bei dieser Gelegenheit nicht versäumen, mich bei den Personen ganz herzlich zu bedanken, die meinen Werdegang begleiteten. Besonders danke ich Studiendirektor Udo Kaden dafür, daß er durch seinen hervorragenden Mathematik- und Physikunterricht die Weichen für meine spätere Studienrichtung stellte. Ebenso möchte ich mich bei den wichtigsten Mentoren während meines Studiums, Prof. Dr. Peter Lesky (em.) und Prof. Dr. Kurt Leichtweiß, für ihre engagierte Förderung während meines gesamten Studiums bedanken. Prof. Dr. Wendelin Degen und Dr. Manfred Oehler danke ich für die ersten Kontakte zur Informatik.

Stuttgart, im Februar 2000 Rolf Rabenseifner

Abstract

Event tracing and monitoring of the program flow and the message exchanges of parallel applications are difficult if each processor has its own unsynchronized clock. A survey of several strategies to generate a global time is given, and the 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. With the control algorithm an approximation of the maximum of all local processor clocks is used as global time. The corrected timestamps can also be used for performance measurements with pairs of events in different processes. A piecewise linear backward amortisation of the clock corrections guarantees a minimal error for measurements of time intervals between events in the same process.

No additional protocol overhead is needed for the new method while tracing the application. The method 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 des Programmflusses sowie des Nachrichtenaustauschs 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 die Grenzen werden aufgezeigt.

Die geregelte logische Uhr, eine neue Methode auf der Basis von Lamports logischer Uhr, wird vorgestellt. Ungenaue Zeitstempel aus Tracefiles werden derart modifiziert, daß sie die Uhrenbedingung erfüllen, d.h. daß der Empfang einer Nachricht einen späteren Zeitstempel als das zugehörige Sendeereignis besitzt. Mit dem Regler wird das Maximum aller lokalen Prozessoruhren als Basis für eine globale Zeit angenähert. Die korrigierten Zeitstempel ermöglichen Leistungsmessungen, bei denen die Ereignisse in verschiedenen Prozessen liegen. Eine stückweise lineare rückwärtige Amortisation der Uhrenkorrekturen garantiert, daß die Fehler bei Messungen von Zeitintervallen zwischen Ereignissen im selben Prozeß minimal sind.

Bei der Erstellung eines Tracefiles ist kein zusätzlicher Protokollaufwand nötig. Die geregelte logische Uhr kann als Filter für Tracefiles implementiert werden. Sie kann aber auch in Monitor- und Debuggingwerkzeuge integriert werden.

Keywords: Logical clock, global time, clock synchronization, backward amortization, causality, monitoring, debugging, trace, distributed systems, timestamps, message-passing.

CRCS Kategorien:

D.1.3 [Concurrent Programming]	Distributed and parallel programming
D.2.5 [Testing and Debugging]	Monitors, Tracing
D.2.6 [Programming Environments]	
D.4.1 [Process Management]	Synchronization
D.4.8 [Performance]	Monitors, Measurements

Inhaltsverzeichnis

Kapitel 1

Einleitung

Eine globale Uhrzeit ermöglicht eine optimale Darstellung von lokalen Ereignissen und Sende-Empfangs-Paaren durch Monitorwerkzeuge zur Analyse paralleler und verteilter Anwendungen.

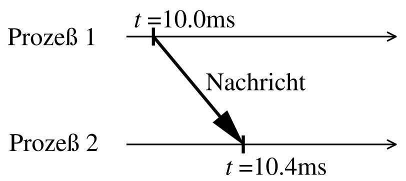


Abb. 1.1: Zeitliniendiagramm basierend auf der physikalischen Zeit

Die Abb. 1.1 zeigt ein Zeitliniendiagramm (timeline diagram) auf der Basis der physikalischen Zeit t (physical time, wall clock time). Die horizontalen Linien sind die Zeitachsen zur Darstellung von Ereignissen in den einzelnen Prozessen eines parallel laufenden Computerprogramms. Die vertikalen Striche symbolisieren Ereignisse (events), hier im Prozeß 1 das Senden einer Nachricht (message) und in Prozeß 2 das Empfangen derselben. Die Nachricht selbst ist mit einem schrägen Vektorpfeil dargestellt. Die horizontale Komponente dieses Vektors repräsentiert die Zeit, die diese Nachricht unterwegs war. Abb. 1.2 zeigt die

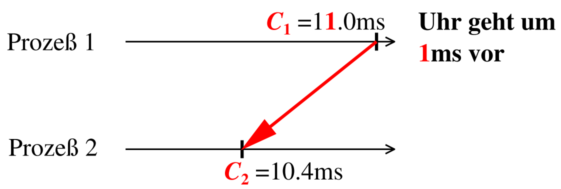


Abb. 1.2: Zeitliniendiagramm mit ungenügend synchronisierten Uhren

gleiche Anwendung, aber basierend auf Zeitstempeln von ungenügend synchronisierten Prozessoruhren C_i. C_1 geht gegenüber der physikalischen Zeit um 1 ms vor, während C_2 korrekt die physikalische Zeit angibt. Das Zeitliniendiagramm zeigt, daß unter solchen Bedingungen bei der Visualisierung des Programmflusses einer parallelen Message-Passing Anwendung die Kausalitäten falsch dargestellt werden, also Nachrichten revers, d.h. mit negativen Nachrichtenlaufzeiten (message delay) gezeichnet werden. Abb. 1.3 enthält die korrekte Darstellung

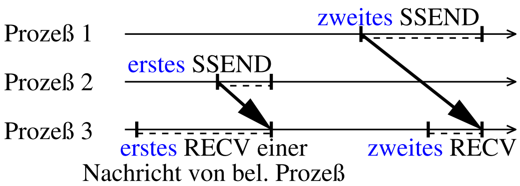


Abb. 1.3: Globales Empfangen mit korrekter Darstellung

eines Programmablaufs, bei dem in den Prozessen 1 und 2 jeweils eine Nachricht mit der Routine MPI_SSEND (synchronous send) des standardisierten Message Passing Interface (MPI) [MPI95] zum dritten Prozeß gesendet wird und diese Nachrichten im dritten Prozeß mit zwei Aufrufen von MPI_RECV(..., MPI_ANY_SOURCE,...) in beliebiger Reihenfolge empfangen werden. Hierbei ist zusätzlich die Dauer der jeweiligen MPI-Routine mit einer horizontalen gestrichelten Linie markiert. Die senkrechten Marken stellen jeweils das Ereignis des Beginns und des Endes der jeweiligen MPI-Routine dar. In Abb. 1.4 wird derselbe Sachverhalt falsch dargestellt, da hier im ersten Prozeß eine nachgehende Uhr zur Zeitmessung benutzt wird. Durch diese nachgehende Uhr werden die Ereignisse im ersten Prozess scheinbar früher dargestellt, und so wird der Anschein erweckt, daß das real zweite Senden nun scheinbar das erste Senden sei. Abb. 1.3 und Abb. 1.4 zeigen also, daß es nicht genügt, nur Rückwärtsbezüge

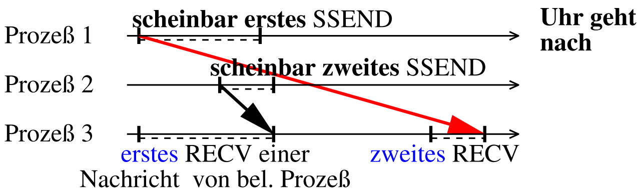


Abb. 1.4: Ereignisse der Abb. 1.3 mit falscher Darstellung

bei der Darstellung von Nachrichten zu verhindern. Durch die nachgehende Uhr in der unteren Darstellung wird der Message-Passing Library unterstellt, daß sie im ersten Aufruf von MPI_RECV die erste Nachricht nicht annimmt, sondern weiter wartet bis das zweite Senden initiiert ist. Beide Beispiele zeigen, daß für das Monitoring paralleler Anwendungen eine für alle Prozesse einheitliche, also globale Zeit, vergleichbar mit der physikalischen Zeit, nötig ist.

Auf Systemen, die keine genügend synchronisierte Prozessoruhren besitzen, d.h. über keine für das Monitoring geeignete globale Uhr verfügen, gibt es verschiedene Strategien, eine solche Uhr zu realisieren. Dieser Bericht stellt hierzu ein neues Verfahren, die geregelte logische Uhr (controlled logical clock), vor. Sie wurde speziell für Workstationcluster oder vergleichbare Systeme entwickelt und kann, wie in Abb. 1.5 auf Seite 11 dargestellt, als Filter für Tracefiles eingesetzt werden. Die geregelte logische Uhr kann als Filterprogramm implementiert werden, das Tracefiles mit logisch fehlerhaften Zeitstempeln einliest, die Zeitstempel korrigiert und die Ereignisinformationen dann mit logisch korrekten Zeitstempeln wieder ausgibt. Hierbei werden die Zeitstempel der eingesetzten Prozessoruhren nachträglich geändert, wenn der Tracefile Rückwärtsbezüge enthält. Rückwärtsbezüge sind hierbei Nachrichten, bei denen das Sendeereignis (send event) einen späteren Zeitstempel als das Empfangsereig-

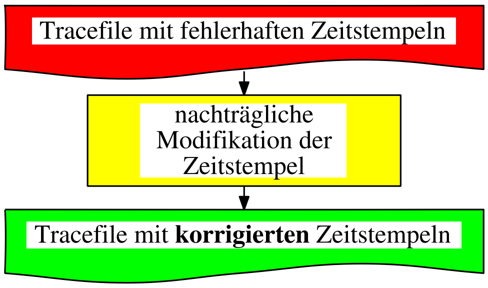


Abb. 1.5: Nachträgliche Korrektur, implementiert als Filter

nis (receive event) besitzt. Die geregelte logische Uhr modifiziert die Zeitstempel, so daß sie Lamports Uhrenbedingung (clock condition) [Lam78] erfüllen, also ein Empfangsereignis dann grundsätzlich einen späteren Zeitstempel als das zugehörige Sendeereignis besitzt.

In Kapitel 2 werden zuerst grundlegende Definitionen bereitgestellt, wie z.B. die Begriffe „synchronisierte Uhren“ und „Uhrenbedingung“. In Kapitel 3 werden verschiedene Anwendungsklassen vorgestellt, und deren Anforderungsprofile abgeleitet. Anschließend werden existierende Verfahren vorgestellt und ihre Grenzen bezüglich des Anforderungsprofils aufgezeigt. In Kapitel 4 wird der Algorithmus der geregelten logischen Uhr definiert. Kapitel 5 enthält eine detaillierte Fehleranalyse zur geregelten logischen Uhr. Kapitel 6 beschreibt Implementierungsdetails des Algorithmus sowie mit dieser Implementierung durchgeführte Tests. In Kapitel 7 wird die geregelte logische Uhr mit seither existierenden Verfahren verglichen und bezüglich des Anforderungsprofils bewertet. Vor diesem Vergleich werde ich das Verfahren von Hofmann und Hilgers [HH98, Hil96] zuerst detailliert vorstellen und anschließend erweitern, da das Originalverfahren von Hofmann und Hilgers für nicht konstante Uhrengeschwindigkeiten ungeeignet ist, und dies aber eine wesentliche Anforderung ist. Kapitel 8 faßt dann die wesentlichen Aspekte dieser Arbeit zusammen. Im Anhang A findet man eine Liste der verwendeten Symbole.

Es ist noch zu bemerken, daß die meisten Graustufenbilder der gedruckten Version dieser Arbeit auch als Farbabbildungen in der Online-Version verfügbar sind.

Kapitel 2

Definitionen

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

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

Definition 2: t bezeichne die physikalische Zeit (real time).
$t(e_i^j)$ mit $t : E \rightarrow \mathbb{R}$ ist die physikalische Zeit des Ereignisses e_i^j.
Eine *Uhr* ist eine Abbildung der physikalischen Zeit auf eine reelle Zahl:
$\mathcal{C} : \{t | t \in \mathbb{R}\} \rightarrow \mathbb{R}$
$C_i(t)$ mit $C_i : \{t | t \in \mathbb{R}\} \rightarrow \mathbb{R}$ sei die *Prozessoruhr* des Prozesses i.
$C(e_i^j) := C_i(t(e_i^j))$, $C : E \rightarrow \mathbb{R}$ ist die *globale Prozessoruhr*.
Es ist hierbei unerheblich, ob die Prozessoruhren als Hardware mit Quarzuhren realisiert sind oder, ob es sich um eine software-technische Abbildung dieser Quarzuhren handelt, die z.B. schon eine grobe Synchronisation der Quarzuhren enthält.
Zur Synchronisation und Präzision von Uhren gibt es in der Literatur mehrere Begriffsbildungen:
Eine Uhr $\mathcal{C} : \{t | t \in \mathbb{R}\} \rightarrow \mathbb{R}$ ist *korrekt* wenn $\underset{t}{\forall} \mathcal{C}(t) = t$ gilt.
Eine Uhr $\mathcal{C} : \{t | t \in \mathbb{R}\} \rightarrow \mathbb{R}$ *geht genau* wenn $\underset{t}{\forall} \frac{d\mathcal{C}(t)}{dt} = 1$ gilt.
$\frac{d\mathcal{C}(t)}{dt}$ ist die *momentane Uhrengeschwindigkeit.*
$\rho(t) = \frac{d\mathcal{C}(t)}{dt} - 1$ ist die *momentane Gangabweichung* (drift) einer Uhr $\mathcal{C} : \{t | t \in \mathbb{R}\} \rightarrow \mathbb{R}$.
$\rho_{\max} = \max_t |\rho(t)|$ ist die *Ganggenauigkeit*[1] (accuracy) der Uhr $\mathcal{C}$.

[1] Im allgemeinen Sprachgebrauch sind die Begriffe Gangabweichung und Ganggenauigkeit Synonyme. Ich verwende sie hier unterschiedlich, um die momentanen Werte und den maximalen Absolutwert sprachlich besser unterscheiden zu können.

Satz 1 *Wenn $\mathcal{C}(t)$ für $t = 0$ korrekt ist (d.h. $\mathcal{C}(0) = 0$), und wenn $\mathcal{C}$ die Ganggenauigkeit $\rho_{\max}$ besitzt, dann gilt:*
$\underset{t\geq 0}{\forall} (1-\rho_{\max})t \leq \mathcal{C}(t) \leq (1+\rho_{\max})t$

Beweis: Der Satz ergibt sich direkt aus den Definitionen durch Integration.

□

Bemerkung 1 *Man sagt auch, daß eine Uhr $\mathcal{C}$ an die physikalische Zeit t mit der Ganggenauigkeit $\rho_{\max}$ gebunden ist, wenn $\underset{t\geq 0}{\forall} (1-\rho_{\max})t \leq \mathcal{C}(t) \leq (1+\rho_{\max})t$ gilt.*

Definition 3: Zwei Uhren $\mathcal{C}_i$ und $\mathcal{C}_k$ sind *intern synchronisiert*, wenn es eine kleine positive Konstante e gibt mit $\underset{t}{\forall}|\mathcal{C}_i(t) - \mathcal{C}_k(t)| \leq e$.

Eine Uhr $\mathcal{C}$ ist *extern synchronisiert*, wenn es eine kleine positive Konstante e gibt mit $\underset{t}{\forall}|\mathcal{C}(t) - t| \leq e$.

Die Kostante e wird als *Synchronisationsfehler* oder auch als *Präzision* (precision) der Synchronisation bezeichnet.

Bemerkung 2 *Die beiden Begriffe* interne *und* externe *Synchronisation lassen sich in einem Modell zusammenfassen. Hierzu führe ich eine* globale Zeit $T : \{t | t \in \mathbb{R}\} \longrightarrow \mathbb{R}$ *ein.*

Definition 4: Eine Uhr $\mathcal{C}$ ist zu einer globalen Zeit T synchronisiert, wenn es eine kleine positive Konstante e gibt mit $\underset{t}{\forall}|\mathcal{C}(t) - T(t)| \leq e$.

Bemerkung 3 *Für $T(t) = t$ ist diese Definition äquivalent mit der Definition von extern synchronisiert. Für $\mathcal{C} = \mathcal{C}_i$ und $T = \mathcal{C}_k$ ist sie äquivalent mit der Definition von intern synchronisiert.*

Definition 5: Ein Uhrensystem $\mathcal{C}_{i,i=1\ldots n}$ heißt *intern synchronisiert*, wenn es eine kleine positive Konstante e' gibt mit $\underset{i,k=1\ldots n}{\forall} \; \underset{t}{\forall}|\mathcal{C}_i(t) - \mathcal{C}_k(t)| \leq e'$

Satz 2 *Wenn die Uhren eines Uhrensystems $\mathcal{C}_{i,i=1\ldots n}$ zur globalen Zeit $T := \mathcal{C}_1$ mit dem Synchronisationsfehler e synchronisiert sind, dann sind sie mit dem Synchronisationsfehler $e' := 2e$ auch intern synchronisiert.*

Beweis:

$$\begin{aligned} |\mathcal{C}_i(t) - \mathcal{C}_k(t)| &= |\mathcal{C}_i(t) - \mathcal{C}_1(t) + \mathcal{C}_1(t) - \mathcal{C}_k(t)| \\ &\leq |\mathcal{C}_i(t) - \mathcal{C}_1(t)| + |\mathcal{C}_1(t) - \mathcal{C}_k(t)| \leq e + e \end{aligned}$$

□

Bemerkung 4 *Zur Analyse der logischen Uhren wird der Begriff* synchronisiert zu einer globalen Zeit T *noch differenziert werden, indem zwischen positiven und negativen Synchronisationsfehlern unterschieden werden wird.*

Definition 6: Für zwei Ereignisse e_k^l, e_i^j gilt die Relation e_k^l *geschah direkt zuvor (happened directly before)* e_i^j, bzw. $e_k^l \tilde{\to} e_i^j$, wenn

(a) $(e_k^l, e_i^j) \in M$, oder (2.1)

(b) die Ereignisse im gleichen Prozeß aufeinanderfolgen, d.h. $k = i \wedge l = j - 1$. (2.2)

Die Relation *geschah zuvor (happened before)*, bzw. $\to$, ist die transitive Hülle der Relation $\tilde{\to}$, d.h. die kleinste Relation, für die zusätzlich gilt:

(c) $e_k^l \to e_i^j \wedge e_i^j \to e_n^m \implies e_k^l \to e_n^m$ (2.3)

Definition 7: Eine Uhr $\mathcal{C} : E \longrightarrow \mathbb{R}$ erfüllt die *Uhrenbedingung*, wenn

$$\underset{e_k^l \in E, e_i^j \in E}{\forall} e_k^l \to e_i^j \implies \mathcal{C}(e_k^l) < \mathcal{C}(e_i^j) \tag{2.4}$$

Unter Vernachlässigung der Gangungenauigkeit der Prozessoruhren läßt sich folgender Satz einfach zeigen:

Satz 3 *Wenn der interne Synchronisationsfehler der Prozessoruhren geringer ist als die kleinste Nachrichtenlaufzeit, dann erfüllen die Prozessoruhren die Uhrenbedingung.*

Eine Präzisierung für theoretisch stetige Uhren mit beschränkter Gangungenauigkeit findet man in [Jez89b, MT95]. In [Hof93b] werden reale Uhren mit diskreten Sprüngen (Uhrentick, Clocktick) untersucht. Dieser Bericht untersucht den Fall, daß die Prämisse dieses Satzes nicht erfüllt ist.

Das Ziel der geregelten logischen Uhr wird sein, daß sie die Uhrenbedingung erfüllt, und daß ihre Ganggenauigkeit hinreichend gut, d.h. in der Regel besser als 5 %, ist. Die geregelte logische Uhr wird eine globale Uhr bilden, die zum Maximum aller Prozessoruhren synchronisiert sein wird.

Kapitel 3

Anwendungsklassen und derzeitige Lösungen

3.1 Anwendungsklassen für eine globale Uhr

In dieser Arbeit betrachte ich Monitoring Werkzeuge, mit denen der Ablauf paralleler und verteilter Programme analysiert werden kann. Im wesentlichen beschränke ich mich auf Programme, deren Synchronisation und Datenaustausch mit explizitem Nachrichtenaustausch (message passing) programmiert ist. Die parallelen Prozesse oder Threads sollen auf Prozessoren laufen, deren Prozessoruhren (C_i) nicht synchronisiert sind, oder deren Synchronisationsfehler größer als die minimale Nachrichtenlaufzeit (minimal message delay) ist. Zur Analyse des Programmablaufs sei das Programm instrumentiert. Die Instrumentierung erzeugt Traceinformationen, während das Programm läuft. Der Trace enthält zu jedem protokollierten Ereignis neben ereignisspezifischen Informationen auch einen Zeitstempel von der jeweiligen Prozessoruhr. Ereignisse sind, z.B. der Start oder das Ende eines Unterprogramms (interne Ereignisse), oder der Beginn des Versendens einer Nachricht (Sendeereignis) oder das Ende des Empfangs einer Nachricht (Empfangsereignis). Bei den Monitoringwerkzeugen sei zwischen zwei Anwendungsklassen unterschieden, den Offline- und den Online-Monitoren. Bei Offline-Monitoren wird die Traceinformation auf ein File geschrieben und nach dem Ende des Ablaufs des zu analysiernden parallelen Programms, wird der Tracefile mittels des Offline-Monitors analysiert. Bei Online-Werkzeugen wird die Traceinformation sofort, bzw. mit für den Betrachter unmerklicher oder tolerierabler Zeitverzögerung, bearbeitet. Die Monitorwerkzeuge sollen eine zeitbasierte Analyse durchführen. Dies sind, z.B. sogenannte Zeitliniendiagramme (time line diagram), wie sie in den Abbildungen 1.1 bis 1.4 skizziert sind, sowie Messungen der Übertragungszeit von Nachrichten oder der Verweildauer in den verschiedenen Unterprogrammen des parallelen Programms auf den einzelnen Prozessoren. Parallele Debugger können derartige Online-Monitore enthalten. Für die Problematik der fehlenden Uhrensynchronisation ist es hierbei unerheblich, ob die Instrumentierung des parallelen Programms software-technisch, hardware-technisch oder mit hybriden Systemen [RTV93] realisiert ist.

3.2 Anforderungen an eine globale Uhr

Die Anforderungen eines Offline-Monitors an eine globale Uhr sind:

(A1) Minimale Fehler bei Zeitdifferenzmessungen zwischen Ereignissen innerhalb eines Prozesses: die Gangabweichung sollte kleiner als 5 % sein.

(A2) Logisch korrekte Zeit, das heißt, daß der Zeitstempel der globalen Uhr eines Emfangsereignisses später (größer) sein muß als der des zugehörigen Sendeereignisses.

(A3) Die logische Korrektheit muß unabhängig von der Qualität der Prozessoruhren sein. Es sollte möglich sein, daß die Uhrengeschwindigkeiten der Prozessoruhren zeitabhängig variieren, z.B. aufgrund einer nicht konstanten Raumtemperatur oder aufgrund einer integrierten internen oder externen Synchronisation der Prozessoruhren, die aber für die logische Korrektheit nicht ausreichend ist.

(A4) Bei Zeitdifferenzmessungen zwischen Ereignissen in verschiedenen Prozessen sollte eine möglichst gute Annäherung an die physikalische Zeitdifferenz erreicht werden.

(A5) Keine oder minimale Störung des Ablaufs der parallelen Anwendung durch die globale Uhr.

(A6) keine oder minimale zusätzliche Rechenzeit der parallelen Anwendung durch den Einsatz einer globalen Uhr, die obige Anforderungen erfüllt.

Bei Online-Monitoren kommt folgende Anforderung hinzu:

(A7) Bei der Darstellung der Traceinformation muß die Zeitverzögerung, die durch die Berechnung der globalen Uhrzeit entsteht, für den Betrachter unmerklich oder noch tolerabel sein. Dies heißt, daß eine derartige Verzögerung möglichst kleiner als eine zehntel Sekunde sein sollte. Diese Anforderung schließt folglich auch mit ein, daß die Darstellung nicht bis zum Abschluß des zu analysierenden Programmlaufs verzögert werden darf.

3.3 Existierende Lösungsansätze

In diesem Kapitel werden existierende Verfahren beschrieben, die auf der Basis nicht oder nicht ausreichend synchronisierter Prozessoruhren eine globale Zeit bilden, die in Monitorwerkzeugen zur Analyse paralleler Anwendungen auf Parallelrechnern oder in Workstation- und PC-Clustern eingesetzt werden kann.

Bei der Darstellung von Ereignissen beim Monitoring, Debugging oder bei Leistungsmessungen gibt es unterschiedliche Ansätze zur Synchronisation der lokalen Uhren. Lamport's diskrete logische Uhr (logical clock) [Lam78] kann direkt zum Monitoring eingesetzt werden [CHK92, ZEA93]. Raynal [Ray87] beschreibt hierzu ein Verfahren zur Begrenzung der Abweichung der logischen Uhren der Prozesse untereinander. Die zu jeder Nachricht zugehörenden Sende- und Empfangsereignisse bilden eine Kausalbeziehung: Das Empfangsereignis kann erst eintreten, wenn zuvor das zugehörige Sendeereignis aufgetreten ist. Die Vektoruhr (vector clock), eine Erweiterung von Fidge [Fid88, Fid89] und Mattern [Mat89], ermöglicht eine äquivalente Darstellung der durch die Sende-Empfangs-Paare gegebenen Kausalbeziehung mittels Monitorwerkzeugen [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 Ereignisse eingeführt. In [PYS92] werden zusätzlich spontane Ereignisse, wie z.B. Kollisionen, berücksichtigt. In der Zusammenfassung [SM94] werden aber auch die Grenzen der diskreten logischen Uhr und der Vektoruhr beleuchtet.

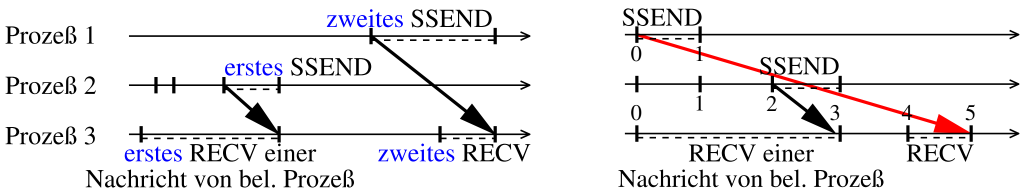


Abb. 3.1: Globales Empfangen dargestellt mittels physikalischer Zeit und diskreter logischer Uhr

Die rechte Darstellung in Abb. 3.1 zeigt eine weitere Beschränkung der Anwendbarkeit von Lamports logischer Uhr auf (s.a. Abb. 1.3 und 1.4 auf S. 10). Hierbei wurden im mittleren Prozeß zu Beginn zwei zusätzliche lokale Ereignisse angenommen. Wie Abb. 1.4 erweckt auch die Darstellung mittels Lamports logischer Uhr den Eindruck, daß das erste globale Empfangen einer Nachricht (d.h. ein Nachrichtenempfang, bei dem nicht festgelegt ist, von welchem sendenden Prozeß die zu empfangende Nachricht kommen soll) die falsche Nachricht empfängt. Dieses Beispiel zeigt damit, daß die logische Uhr 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 Bestimmung der Gangabweichungen [Dun91, Dun94], oder besser am Anfang und Ende [MT95] mit einem linearen Ausgleich während der Dauer der Anwendung, sind die in softwarebasierten Monitorwerkzeugen [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 Ressourcenbelastung, wie z. B. mit xntp [Mil92], ist normalerweise aufgrund der stark variierenden Nachrichtenlaufzeiten in einem lokalen Netz nicht ausreichend exakt, d.h. der Sychronisationsfehler kann größer sein als die minimale Nachrichtenlaufzeit.

Eine Alternative ist die tracebasierte Synchronisierung. Auf der Basis eines Tracefiles und der Prämisse, daß ein Empfangsereignis frühestens um die minimale Nachrichtenlaufzeit nach dem Sendeereignis ankommen darf, werden die Uhrenabweichungen ermittelt und nach Beendigung des zu monitorenden Programms wird die Traceinformation analysiert, und werden die Zeitstempel modifiziert, damit die modifizierten Zeitstempel die im letzten Kapitel definierten Anforderungen möglichst erfüllen.

Duda et al. [DHHB87] entwickelten hierzu zwei Verfahren. Das erste Verfahren enthält eine Regressionsanalyse, die aber nicht ausreicht, um eine korrekte Darstellung von Kommunikationsbeziehungen zu erreichen. Beim zweiten Verfahren werden zwei konvexe Hüllen gebildet. Es ist in Abb. 3.2 auf Seite 18 dargestellt. Das Verfahren bestimmt den Synchronisationsfehler zwischen zwei Prozessoruhren C_i und C_k als eine lineare Funktion der Prozessorzeit. Zur Vereinheitlichung der Darstellung wurde in Abb. 3.2 für die Berechnung dieser linearen Ausgleichsfunktion die physikalische Zeit t als Abszisse gewählt. Dies ist aber unerheblich, da der Maßstab der Abszisse den Unterschied zwischen den Prozessor-Zeiten der

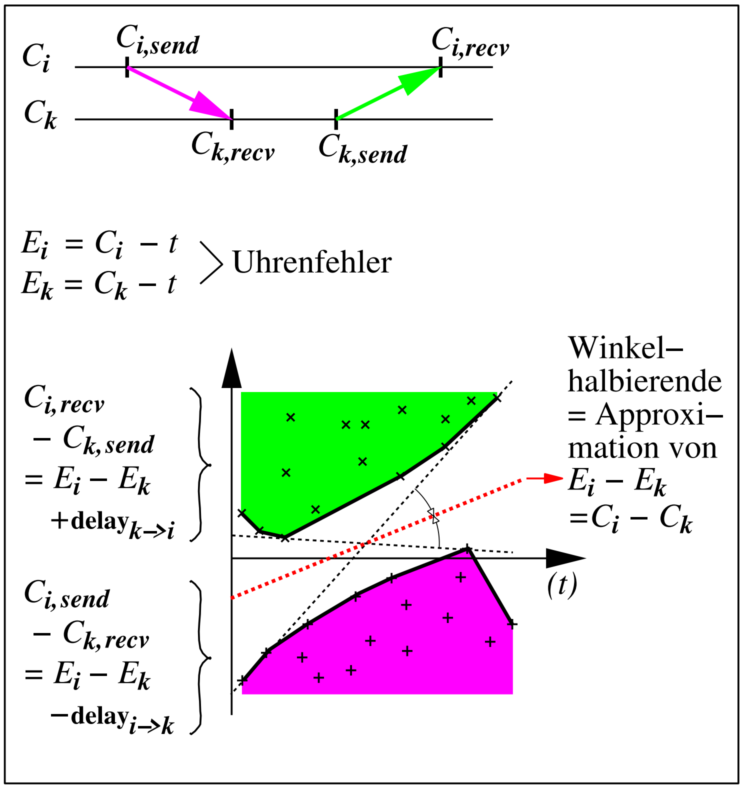


Abb. 3.2: Algorithmus von Duda

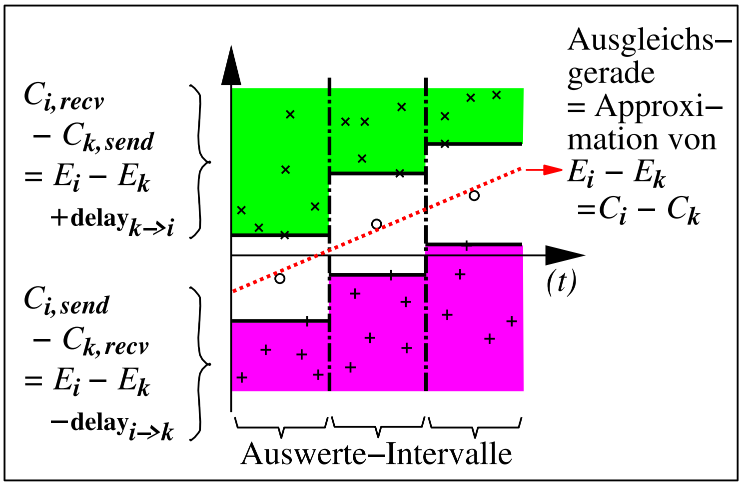


Abb. 3.3: Algorithmus von Hofmann

beteiligten Prozessoren im allgemeinen nicht auflösen kann. Zur Bestimmung der Synchronisationsfehler werden hierzu in einem Diagramm über der Zeit die Differenzen $C_{i,recv} - C_{k,send}$ aller Nachrichten von Prozeß k nach i und die Differenzen $C_{i,send} - C_{k,recv}$ aller Nachrichten von Prozeß i nach k aufgetragen. Die ersteren Differenzterme sind mit den $\times$ Kreuze in der oberen unterlegten Fläche, die zweiten Differenzterme sind mit den + Kreuze in der unteren Fläche dargestellt. Anschließend werden die konvexen Hüllen der beiden Punktemengen gebildet. Dies sind nun die Ränder der beiden unterlegten Flächen. Anschließend werden die beiden Geraden mit der größten und der kleinsten Steigung zwischen den beiden Gebieten ermittelt. Die Uhrendifferenz wird dann linear mittels der Winkelhalbierenden der beiden Geraden approximiert. Jézéquel [Jez89a] erweiterte diese Verfahren für eine beliebige Prozessor-Topologie mittels eines *minimal spanning tree*[1] Algorithmus.

Hofmann [Hof93a] vereinfachte Dudas Verfahren, indem er mittels einer Minimum/Maximum-Bildung auf Teilintervallen jeweils zeitlich konstante Uhrendifferenzen ermittelt und anschließend durch eine Regressionsanalyse über alle Teilintervalle die anfängliche Uhrenabweichung und die (konstante) Uhrengeschwindigkeit approximiert. Abb. 3.3 auf Seite 18 skizziert den Algorithmus. Das Ergebnis ist eine Approximation der Uhrendifferenzen in Abhängigkeit von der Zeit. In dem unteren Diagramm werden die Uhrendifferenzen abzüglich (+ Kreuze), bzw. zuzüglich ($\times$ Kreuze) der Nachrichtenlaufzeiten der einzelnen Nachrichten, über der (physikalischen) Zeit eingetragen. Anschließend wird innerhalb der einzelnen (hier drei) Teilintervalle der punktfreie Parallelstreifen dazwischen gebildet. Eine Ausgleichsgerade durch die Mittelpunkte dieser Streifen approximiert die Uhrendifferenz in Abhängigkeit der Zeit. Hofmann und Hilgers [HH98, Hil96] vereinfachen die von Duda und Jézéquel angegebenen Verfahren zur Lösung des Mehrprozessor-Problems, indem sie das Problem für die konstanten Uhrenabweichungen innerhalb der Teilintervalle auf ein graphentheoretisches Verfahren zur Bestimmung kürzester Wege reduzieren.

Babaoğlu und Drummond [BD87, DB93] zeigen, daß eine fast kostenlose Synchronisierung möglich ist, wenn die Anwendung hinreichend oft einen vollständigen Nachrichtenaustausch zwischen den Prozessen durchführt. Einen Überblick über weitere Arbeiten findet man in [YM93]. Die Grenzen dieser Methoden sind bestimmt durch die Varianz der Nachrichtenlaufzeiten, den nicht linearen Zusammenhang von Nachrichtenlaufzeit und Nachrichtenlänge, sowie durch die teilweise einseitige Kommunikationstopologie von Anwendungen (z.B. Producer/Consumer Szenarien).

[1] Jézéquel bezeichnete den Algorithmus als spawning tree, in der Literatur ist aber spanning tree üblicher, vgl. [Bol79], pp. 7-11.

3.4 Kategorisierung und Grenzen der existierenden Lösungsansätze

Die vorgestellten Verfahren lassen sich generell in zwei Kategorien einteilen:

1) Synchronisation der Prozessoruhren und

2) die nachträgliche Modifikation der Zeitstempel in der Traceinformation, bevor die Traceinformation von einem Monitorwerkzeug weiterverarbeitet wird.

Die Verfahren zur Synchronisation der Prozessoruhren wiederum untergliedern sich in:

1a) ressourcenarme Synchronisationsverfahren, die kontinuierlich während der Laufzeit des zu analysierenden parallelen Programms durchgeführt werden. Diese Verfahren sind im allgemeinen nicht ausreichend, wenn sie zur Synchronisation der Prozessoruhren die gleichen Nachrichtenaustauschmethoden nutzen, die auch zur Kommunikation von den zu analysierenden Programmen benutzt werden, da in diesem Fall nicht garantiert werden kann, daß der interne Synchronisationsfehler geringer ist als die minimale Nachrichtenlaufzeit ist, und daher die Prämisse von Satz 3 nicht erfüllt ist, und somit die Anforderung (A2) aus Kapitel 3.2, d.h. die Uhrenbedingung, nicht erfüllt ist.

1b) Synchronisationsverfahren, die am Anfang und/oder am Ende der Laufzeit des zu analysierenden parallelen Programms die Prozessoruhren mit großer Genauigkeit synchronisieren und während der Laufzeit des zu analysierenden Programms die Prozessoruhren linear in der Zeit korrigieren. Im Falle der Synchronisation am Anfang *und* Ende kann die lineare Korrektur nur als nachträgliche Zeitstempel-Korrektur (Kategorie 2) durchgeführt werden. Bei der typischen Ganggenauigkeit von Quarzuhren von 10^{-5} bis 10^{-6} und einer minimalen Übertragungsdauer von 200 μs genügen nach einer Anfangssynchronisation mit konstanter Korrektur der Prozessoruhren schon 20 bis 200 Sekunden, bis der Synchronisationsfehler die minimale Übertragungsdauer überschreiten kann. Bei modernen Kommunikationsnetzen liegen die Nachtrichtenlaufzeiten unter 10 μs, d.h. schon nach ein bis zehn Sekunden Programmlaufzeit ist ein zu großer Synchronisationsfehler aufgrund unterschiedlicher Uhrengeschwindigkeiten möglich. Maillet und Tron zeigen in [MT95], daß auch eine Bestimmung der Gangabweichungen vor dem Start der Anwendung (und entsprechende lineare Korrektur der Zeitstempel) bei längeren Laufzeiten des zu analysierenden Programms (z.B. 2 Stunden) keine ausreichende Korrektur ermöglicht. Außerdem verzögert eine derartige Bestimmung der Gangabweichungen den Start der Anwendung im Minuten-Bereich. Maillet und Tron zeigen aber auch, daß die Grenzen der Synchronisation am Anfang und Ende der Laufzeit (und mit linearer Korrektur der Zeitstempel dazwischen) bei einem Synchronisationsfehler von etwa 150 μs liegen, d.h. bei Kommunikationsnetzwerken mit Datenübertragungszeiten geringer als 150 μs kann eine logisch korrekte globale Uhr mit diesem Synchronisationsverfahren nicht mehr gewährleistet werden. Auch wenn Maillet und Tron diese Fehlerrate nur an einem Beispiel zeigen, so kann es doch verallgemeinert werden: Im allgemeinen Fall darf man nicht annehmen, daß ein geringerer Synchronisationsfehler garantiert werden könnte, als der in diesem Fallbeispiel aufgezeigte interne Synchronisationsfehler. Grundsätzlich ist die Synchronisation am *Anfang und Ende* des Programmlaufs exakter und schneller durchführbar als eine Bestimmung der *Uhrendifferenzen und Gangabweichungen* vor dem Programmstart. Die Bestimmung am Anfang und Ende kann aber nur bei Offline-Monitoren angewandt werden, da sie die Anforderung (A7) aus Kapitel 3.2 nicht erfüllt.

Prinzipiell können die Verfahren mit Anfangs- und/oder Ende-Synchronisation noch dadurch verbessert werden, daß die zur Synchronisation durchgeführten Messungen nicht auf den einzelnen Prozessoren ausgewertet werden, sondern ebenfalls auf den Tracefile geschrieben werden und nachträglich *zentral* ausgewertet werden. Durch die zentrale Auswertung kann eine bessere Synchronisation, d.h. mit einem geringeren Synchronisationsfehler, erreicht werden.

Bei der nachträglichen Korrektur der Zeitstempel der im Tracefile aufgezeichneten Ereignisse lassen sich die im vorherigen Kapitel vorgestellten Verfahren in zwei Unterkategorien einordnen:

2a) Die erste Unterkategorie sind Algorithmen, die zuerst auf der Basis der Traceinformation nachträglich die Synchronisationsfehler ermitteln und anschließend die Zeitstempel korrigieren. Diese Unterkategorie sei mit *tracebasierter Synchronisation* bezeichnet. Meilensteine in der Entwicklung dieser Verfahren sind die Arbeiten von Duda et al. [DHHB87], Jézéquel [Jez89a], Hofmann [Hof93a] und Hofmann und Hilgers [HH98, Hil96]. Diese Verfahren erfüllen alle Anforderungen, außer den Anforderungen (A3) und (A7), das heißt, daß die logische Korrektheit nur bei hinreichend konstanter Gangabweichung realisiert wird und, daß diese Verfahren nur bei Offline-Monitoren eingesetzt werden können, da es sich um Algorithmen mit zwei Durchläufen (two pass) handelt, die daher nicht als Filter eingesetzt werden können. Bei den Verfahren von Hofmann und Hilgers zeige ich auch in einer detaillierten Analyse in Kapitel 7, daß auch die Anforderung (A2) (logische Korrektheit) nicht immer erfüllt ist, auch wenn die Ausgangsdaten eine lineare Korrektur ermöglichen, und diese auch von Dudas Algorithmus korrekt durchgeführt wird.

2b) Die zweite Kategorie sind Verfahren, die auf Lamports logischer Uhr [Lam78] basieren. Die ganzzahlige logische Uhr von Lamport erfüllt zwar die Uhrenbedingung (Anforderungen A2 und A3), aber sie ist für prozeßlokale und globale Zeitmessungen (Anforderungen A1 und A4) nicht geeignet. Die Erweiterung dieser Methode zur Vektoruhr, gleichzeitig entwickelt von Fidge [Fid88, Fid89] und Mattern [Mat89], ändert an diesem Mangel nichts. Die Erweiterung von Lamports ganzzahlig logischer Uhr zu einer stückweise differenzierbaren logischen Uhr LC, deren Uhrengeschwindigkeit stückweise wie die der Prozessoruhren C_i ist (d.h. $dLC_i/dt := DC_i/dt$, siehe [Jez89a]), verbesssert zwar die Korrektheit und Genauigkeit von Zeitmessungen, aber die Anforderungen (A1) und (A4) sind bei weitem nicht erfüllt, da sich diese logische Uhr bei Empfangsereignissen beliebig weit vorstellt, um die Uhrenbedingung zu erfüllen. In Kapitel 4.4.1 wird ein Beispiel gezeigt, das beliebig große absolute Synchronisationsfehler aufweist. Das sprunghafte Vorstellen der logischen Uhr bei Empfangsereignissen, deren Zeitstempel zu gering ist, impliziert, daß die Gangabweichung hier unendlich ist, bzw. beliebig groß ist, wenn zwei zu messende Ereignisse nur hinreichend nahe beieinander liegen, und das eine vor und das andere nach dem Uhrensprung liegt. Die Anforderungen (A5) und (A6), d.h. die Forderung nach einer minimalen Störung des zu analysiernden Programms, sind – wie bei allen Verfahren der Kategorie 2 – erfüllt, da die Korrektur der Zeitstempel nicht innerhalb des zu analysierenden Programms durchgeführt wird. Die Anforderung (A7), d.h. die Eignung für Online-Monitore, ist ebenfalls erfüllt, da Lamports logische

Uhren zur Berechnung einer Zeitstempelkorrektur nur Daten benötigen, die, bezogen auf die physikalische Zeit, früher entstanden sind.

3.5 Die Idee der geregelten logischen Uhr

Die geregelte logische Uhr ist eine Erweiterung von Lamports stückweise differenzierbarer logischer Uhr, bei der das Problem der beliebig großen Synchronisationsfehler durch eine *geregelte Verlangsamung* der logischen Uhren und das Problem des sprunghaften Vorstellens durch eine *rückwärtige stückweise lineare Amortisation* gelöst wird. Die geregelte logische Uhr ist eine neuartige Weiterentwicklung von Lamports logischer Uhr 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 Debuggingwerkzeuges keinen zusätzlichen Protokolloverhead benötigt, sondern eine nachträgliche Synchronisation anhand der vorhandenen Zeitstempel und Kommunikationsbeziehungen durchführt. Im Gegensatz zur tracebasierten Synchronisation kann die geregelte logische Uhr auch bei Uhren, deren Granularität größer als die minimale Nachrichtenlaufzeit ist, angewandt werden.

Wesentliche Vorteile der geregelten logischen Uhr gegenüber den seither bekannten Lösungen sind:

a) Die geregelte logische Uhr kann bei variierenden Uhrendifferenzen eingesetzt werden, d.h. daß praktisch keine Anforderungen an die Konstanz der Uhrengeschwindigkeiten gestellt werden.

b) Sie kann als One-Pass-Algorithmus implementiert werden, d.h. daß, wie in Abb. 1.5 auf Seite 11 dargestellt, die geregelte logische Uhr als Filter für Tracefiles programmiert werden kann, das Tracefiles mit logisch fehlerhaften Zeitstempeln einliest und die Ereignisinformationen dann mit den korrigierten und logisch korrekten Zeitstempeln wieder ausgibt.

Letzteres bedeutet bei einer stärkeren Zunahme der Rechengeschwindigkeiten im Vergleich zur Zunahme der Geschwindigkeiten des Plattenzugriffs, daß die geregelte logische Uhr auch in der Ausführungszeit schneller ist als die seitherigen Ansätze. Während die im letzten Kapitel genannten Verfahren meist eine Approximation des *Durchschnitts* der Prozessoruhrzeiten als globale Zeit verwenden, wird bei der geregelten logischen Uhr eine Approximation des *Maximums* realisiert. Diese Arbeit ist eine Erweiterung der Arbeiten [Rab96, Rab97] des Autors und erzielt wesentlich geringere Fehler und bedarf nicht mehr der in diesen Arbeiten angegebenen Beschränkung der Uhrenabweichungen auf das 4-fache der minimalen Nachrichtenlaufzeit.

Im folgenden Kapitel wird der Algorithmus der geregelten logischen Uhr entwickelt und definiert.

Kapitel 4

Die geregelte logische Uhr

Da *Lamports logische Uhr* die Basis dieser Arbeit ist, wird sie im folgenden Unterkapitel kurz vorgestellt. Anschließend wird eine *einfache logische Uhr* definiert, die zwar schon einen Bezug zur Prozessorzeit (und damit zur physikalischen Zeit) besitzt, die aber die gestellten Anforderungen noch nicht erfüllt. Aufbauend auf dieser Uhr wird dann in den Unterkapiteln 4.3 und 4.4 die *geregelte logische Uhr* und ihr Regler definiert. In Kap. 4.5 wird der Algorithmus noch durch die *rückwärtige Amortisation* ergänzt.

4.1 Lamports logische Uhr

C_i sei die Prozessoruhr des Prozesses i.

Algorithmus 1: Lamports logische Uhr $\boldsymbol{LC_i^{int}}$ ist definiert mit

$$LC_i^{int}(e_i^j) := \begin{cases} 0 \quad \text{für } j = 0 \\ LC_i^{int}(e_i^{j-1}) + 1 \quad \text{für interne und Sendeereignisse} \\ \max(LC_i^{int}(e_i^{j-1}) + 1, LC_k^{int}(e_k^l) + 1) \quad \text{wenn } \underset{e_k^l}{\exists} (e_k^l, e_i^j) \in M, \\ \qquad \text{d.h. wenn } e_i^j \text{ ein Empfangsereignis ist und } e_k^l \text{ das} \\ \qquad \text{zugehörige Sendeereignis ist.} \end{cases} \tag{4.1}$$

Die globale logische Uhr ist definiert als

$$LC^{int}(e_i^j) := LC_i^{int}(e_i^j).$$

Abb. 4.1 zeigt ein Beispiel. In dem Diagramm rechts oben sind die Ereignisse in den beiden Prozessen 1 und 2 als kleine Kreise dargestellt. Auf der Abszisse ist ihre physikalische Zeit und auf der Ordinate ist jeweils die ihnen zugehörige Prozessorzeit C_1 oder C_2 zugeordnet. Gestrichelt dargestellt sind jeweils die Abbildungen $t \mapsto C_1(t)$ und $t \mapsto C_2(t)$, d.h. die Ereignisse des Prozesses 1 liegen auf der C_1-Linie und die des Prozesses 2 auf der C_2-Linie. Die schrägen Pfeile symbolisieren Nachrichten, z.B. die erste Nachricht geht von dem mit „1“ markierten Sendeereignis in Prozeß 1 zu dem mit „2“ markierten Empfangsereignis in Prozeß 2. Die Zahlen, mit denen die Ereignisse markiert sind, sind die Werte LC_1 und LC_2 von Lamports logischer Uhr gemäß Gleichung (4.1) in Algorithmus 1.

Bei den Ereignissen mit den Werten $LC_2^{int} = 2$ und $LC_1^{int} = 7$ (eingekreiste Werte) war der rechte Term in (4.1) maßgeblich, d.h., daß die logische Uhr zusätzlich vorgestellt werden mußte, damit keine Rückwärtsbezüge auftreten.

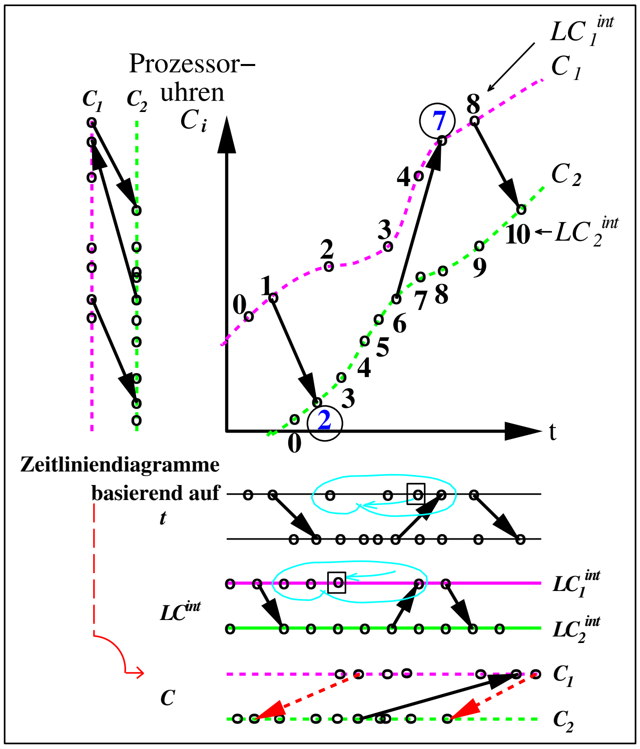


Abb. 4.1: Lamports logische Uhr

Links neben dem beschriebenen t-C_i-Diagramm befindet sich das zugehörige Zeitliniendiagramm auf der Basis der Prozessoruhren. Dies heißt, daß die vertikalen Linien die Zeitachsen C_1 (links) und C_2 (rechts) darstellen. Die Richtung der fortschreitenden Zeit ist nach oben. Es handelt sich hierbei um eine horizontale Parallelprojektion der Ereignisse auf die vertikalen Zeitachsen C_i. Auf den Zeitachsen sind die Ereignisse jeweils zu ihrem Zeitpunkt C_i wieder als kleine Kreise eingezeichnet. Die Pfeile stellen wieder die Nachrichten dar. Man sieht, daß die erste und dritte Nachricht jeweils mit einem rückläufigen Pfeil abgebildet ist. Diese Darstellung ist logisch falsch, da sie den Eindruck erweckt, Nachrichten zu empfangen, bevor sie überhaupt abgeschickt worden sind.

Das unterste horizontale Diagramm stellt nochmals dieses Zeitliniendiagramm auf der Basis der Prozessoruhren dar. Zur Verdeutlichung sind die logisch falsch dargestellten Nachrichten hier gestrichelt dargestellt.

Das oberste horizontale Zeitliniendiagramm stellt den korrekten Verlauf der Ereignisse auf der Basis der physikalischen Zeit t dar. Hier handelt es sich um eine vertikale Parallelprojektion der Ereignisse auf die horizontalen physikalischen Zeitachsen (t) in den beiden Prozessen.

Das mittlere horizontale Zeitliniendiagramm basiert auf Lamports logischer Uhr. Die Ereignisse sind gemäß ihren Werten LC_i^{int} auf den Zeitlinienachsen eingetragen. Lamports logische Uhr erfüllt die Uhrenbedingung, d.h. Rückwärtsbezüge (die gestrichelten Pfeile im untersten, auf den Prozessoruhren basierten Zeitliniendiagramm) werden verhindert, wie das mittlere Zeitliniendiagramm zeigt. Lamports logische Uhr eignet sich aber nicht für Leistungsmessungen, da sie keinen Bezug zur physikalischen Zeit besitzt.

Die folgenden Betrachtungen dienen zur Veranschaulichung der Anforderungen an Uhren, die für Monitorwerkzeuge eingesetzt werden können. Da Lamports logische Uhr nicht für diese Anforderungen entworfen ist, kann diese Uhr als Beispiel dafür dienen, daß eine logische Uhr zwar die Uhrenbedingung erfüllt, aber den nachfolgend genannten, für ein Monitorwerkzeug wichtigen, Anforderungen nicht genügt. Eine der Anforderungen ist, daß Leistungsmessungen möglich sind, also ein Bezug zur physikalischen Zeit besteht. Bei Lamports logischer Uhr fehlt dieser Bezug. Das sieht man deutlich an den vier mit einer punktierten Linie eingekreisten Ereignissen: Das mit einem Rechteck markierte Ereignis hat in Wirklichkeit (d.h. bezüglich der physikalischen Zeit t, dargestellt im obersten Zeitliniendiagramm) einen etwa gleichgroßen zeitlichen Abstand zu seinem vorhergehenden und zu seinem nachfolgenden Ereignis. In dem auf Lamports logischer Uhr basierenden mittleren Diagramm hingegen ist dieses Ereignis dreimal soweit von seinem nachfolgenden Ereignis im Vergleich zu seinem vorhergehenden Ereignis entfernt.

Außerdem können auch globale Wartepunkte (z.B. `MPI_RECV(...MPI_ANY_SOURCE...)`, [MPI95]), wie schon in Abb. 3.1 auf Seite 17 aufgezeigt, mit Lamports logischer Uhr nicht adäquat dargestellt werden. Der Schwerpunkt von Lamports logischer Uhr besteht darin, daß die Uhrenbedingung erfüllt ist.

4.2 Die einfache logische Uhr

Die einfache logische Uhr ist eine Erweiterung von Lamports logischer Uhr. Der Name *einfach* wurde zur Unterscheidung von der darauf aufbauenden *geregelten* logischen Uhr gewählt.

Definition 8: t sei die physikalische Uhrzeit, $T(t)$ sei eine globale Zeit, zu der die Prozessoruhren $C_i(t)(i = 1..n)$ mit einer begrenzten Abweichung synchronisiert sind, d.h. es existieren Konstanten $-e_i^-$ und e_i^+ mit $-e_i^- \leq C_i(t) - T(t) \leq e_i^+$.

Die einfache logische Uhr LC wird folgendermaßen gebildet: Normalerweise läuft sie wie C_i, d.h. $LC_i := C_i$. Beim Empfangen eines Ereignisses wird sie auf $\max(LC_i, \mu + LC_k$ des Senders beim Absenden) vorgestellt, mit $\mu =$ minimale Nachrichtenlaufzeit, die z.B. bei einer internen Synchronisation vor Beginn der Anwendung ermittelt werden kann. Wenn die logische Uhr auf der Basis dieser Regel vorgestellt worden ist, dann bleibt sie anschließend nahezu stehen, solange sie gegenüber der lokalen Prozessoruhr noch vorgeht, d.h. solange $LC_i > C_i$. Genaugenommen wird sie hierbei jeweils bei jedem Ereignis um ein kleines δ vorgestellt. Es handelt sich hierbei um eine Modifikation der von Lamport [Lam78] entwickelten logischen Uhr.

Algorithmus 2: Die exakte Definition der **einfachen logischen Uhr *LC*** ist:

$$LC_i(e_i^j) := \begin{cases} \max(LC_k(e_k^l) + \mu_{k,i},\ LC_i(e_i^{j-1}) + \delta_i,\ C_i(t(e_i^j))) & (4.2) \\ \quad \text{wenn} \underset{e_k^l}{\exists} (e_k^l, e_i^j) \in M & \\ \max(LC_i(e_i^{j-1}) + \delta_i,\ C_i(t(e_i^j))) \quad \text{andernfalls,} & (4.3) \\ \text{wobei die Terme } LC_i(e_i^{j-1}) + \delta_i \text{ bei } j = 0 \text{ entfallen} & \end{cases}$$

mit

$\delta_i =$ minimaler Abstand zweier Ereignisse in Prozeß i, d.h. δ_i sind Konstanten für die gelten:

$$\delta_i > 0 \quad \wedge \quad \underset{i,j}{\forall}\, T(t(e_i^j)) - T(t(e_i^{j-1})) \geq \delta_i \tag{4.4}$$

$\mu_{k,i}$ = minimale Laufzeit einer Nachricht von Prozeß k nach Prozeß i, d.h. $\mu_{k,i}$ sind Konstanten für die gelten:

$$\mu_{k,i} > 0 \quad \wedge \quad \underset{(e_k^l, e_i^j) \in M}{\forall} T(t(e_i^j)) - T(t(e_k^l)) \geq \mu_{k,i} \tag{4.5}$$

Die zugehörige globale logische Uhr ist definiert als

$$LC(e_i^j) := LC_i(e_i^j) \tag{4.6}$$

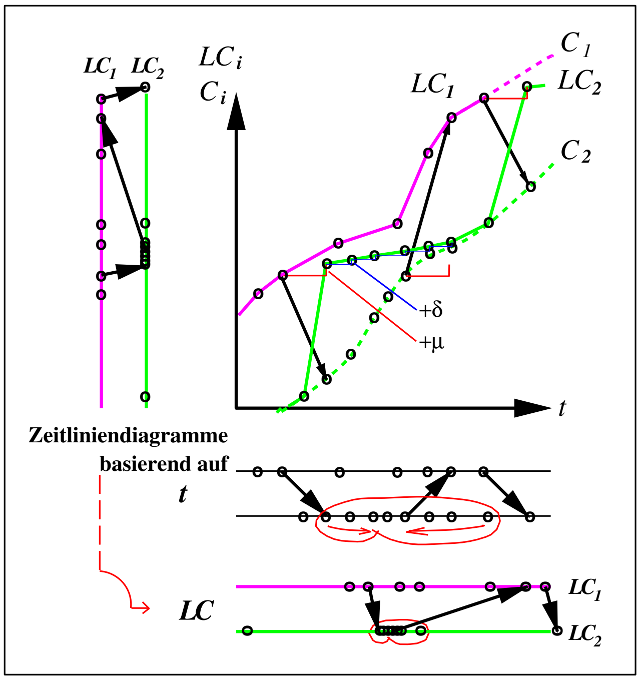


Abb. 4.2: Einfache logische Uhr

In Abb. 4.2 wird der Algorithmus anhand des vorhergehenden Beispiels veranschaulicht. In dem Diagramm rechts oben sind wieder die Ereignisse mit ihrer Prozessorzeit C_i über der physikalischen Zeit t aufgetragen. Zusätzlich wurden die Ereignisse auch mit ihren Werten LC_1 und LC_2 dargestellt und mit durchgezogenen Linien verbunden. Da im Falle des Prozesses 1 die Werte von LC_1 und C_1 identisch sind, wurde auf die gestrichelte Darstellung von C_1 im Bereich der durchgezognenen stückweise linearen Darstellung[1] von LC_1 verzichtet. Im folgenden sei der Verlauf der einfachen logischen Uhr LC_2 erläutert: LC_2 beginnt beim ersten Ereignis in Prozeß 2 mit dem Wert der Prozessoruhr C_2. Das zweite Ereignis ist ein Empfangsereignis. Hier wird der Term $LC_k(e_k^l) + \mu_{k,i}$ in (4.2) relevant, da dieser Wert den der eigenen Prozessoruhr $C_i(t(e_i^j))$ übersteigt. Dies wird im Diagramm durch die mit $+\mu$ markierten Stufe angezeigt. Da nun der Wert der von LC_2 den Wert von C_2 übersteigt, wird bei den folgenden internen Ereignissen der Term $LC_i(e_i^{j-1}) + \delta_i$ in (4.2) relevant. Dies wird

[1]Die logischen Uhren sind nur für Ereignisse im Tracefile definiert. Zur besseren Veranschaulichung wurden die Punktmengen $\{(LC_i(e_i^j), t(e_i^j))\}_{j=1,\ldots}$ für jeden Prozeß P_i mit geraden Linien zu stückweise linearen Kurven verbunden.

im Diagramm durch die mit $+\delta$ markierte Stufen angezeigt. Dadurch steigt der Wert von LC_2 nur noch schwach an, bis er beim vorletzten Ereignis wieder den Wert von C_2 erreicht hat. Der Term $C_i(t(e_i^j))$ in (4.2) sorgt dafür, daß LC_2 niemals unter C_2 sinken kann. Nun wiederholt sich durch die dritte Nachricht der Vorgang wieder, d.h. LC_2 wird wieder auf den Wert $LC_k(e_k^l) + \mu_{k,i}$ vorgesetzt. Es bleibt noch zu bemerken, daß die zweite Nachricht, die zur ersten gegenläufig ist, keine Auswirkungen auf die einfache logische Uhr hat, da hier die beim Sendeereignis angefügte Treppe $+\mu$ zu einem geringeren Wert als dem Wert der Prozessoruhr C_1 des Empfangsereignisses führt. Daher bleibt hier LC_1 bei dem Wert von C_1.

Links von dem t-LC_i-Diagramm ist mittels Projektion das Zeitliniendiagramm zur einfachen logischen Uhr LC dargestellt. Es ist als horizontales Diagramm am unteren Rand des Bildes wiederholt. Wenn man die beiden horizontalen Zeitliniendiagramme vergleicht, sieht man bei den eingekreisten Ereignissen, daß sich die einfache logische Uhr zur Darstellung in Monitorwerkzeugen noch nicht eignet, da sie weiter auseinander liegende Ereignisse nach einer Uhrenkorrektur sehr eng zusammenführt, da diese dann nur noch durch δ_i getrennt dargestellt werden.

Satz 4 *Die einfache logische Uhr LC erfüllt die* **Uhrenbedingung** *(clock condition).*

Beweis: Der Algorithmus 2 genügt Lamports Regeln IR1[2] und IR2[3] in [Lam78] und erfüllt damit die Uhrenbedingung.

□

Damit die einfache logische Uhr die Uhrenbedingung erfüllt, ist die Qualität der Prozessoruhren irrelevant. Die Uhrenbedingung ist also auch erfüllt, wenn z.B. die Prozessoruhren zwischen zwei Ereignissen stehen bleiben, ja sogar wenn sie rückwärts gehen würden. Um dies zu erreichen, wurde im Algorithmus der einfachen logischen Uhr der Term δ_i eingeführt, der mindestens zwischen zwei Ereignissen zur einfachen logischen Uhr hinzu addiert wird. Im folgenden wird der Fehler ausgehend von C_i modelliert:

$$\underset{e_i^j}{\forall}\, t_0 < t(e_i^j) < t_e \;\wedge \underset{t\,:\,t_0<t<t_e}{\forall} -e^- \leq -e_i^- \leq C_i(t) - T(t) \leq e_i^+ \leq e^+ \tag{4.7}$$

d.h. die Prozessoruhren gehen gegenüber der globalen Zeit um maximal e^-, bzw. e_i^- nach, und um maximal e^+, bzw. e_i^+ vor.

Satz 5 *Wenn alle Uhren C_i nie mehr als e^+ vorgehen, dann geht auch die einfache logische Uhr LC nie um mehr als e^+ vor, d.h.*

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

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

$$LC(e_i^j) - T(t(e_i^j)) > e^+ \tag{4.8}$$

[2] IR1. Jeder Prozeß P_i inkrementiert seine Uhr C_i [hier LC_i, Übertragung in die Nomenklatur dieser Arbeit] grundsätzlich zwischen zwei aufeinanderfolgenden Ereignissen.

[3] IR2. (a) Wenn ein Ereignis a das Absenden einer Nachricht m in Prozeß P_i ist, dann wird der Nachricht der Zeitstempel $T_m := C_i(a)$ [hier $LC_i(a)$] zugewiesen. (b) Beim Empfangen der Nachricht m im Prozeß P_j [hier P_k], setzt Prozeß P_j seine Uhr C_j [hier LC_k] größer oder gleich seinem augenblicklichen Wert und größer als T_m. [Dies muß für jedes Empfangsereignis gelten.]

und o. B. d. A. sei e_i^j das früheste derartige Ereignis. (4.9)

Fall a) $LC(e_i^j) = LC_i(e_i^j)$ sei definiert mittels (4.2). Dann gilt:

$$\begin{aligned} LC(e_i^j) - T(t(e_i^j)) &\overset{(4.6)}{=} LC_i(e_i^j) - T(t(e_i^j)) \\ &\overset{(4.2)}{=} \max(LC_k(e_k^l) + \mu_{k,i},\, LC_i(e_i^{j-1}) + \delta_i, \\ &\qquad C_i(t(e_i^j)))\ -\ T(t(e_i^j)) \end{aligned}$$

und (α) falls beim Maximum in (4.2) der erste Fall zutrifft:

$$\overset{(4.2,1.\text{Ausdruck})}{=} LC_k(e_k^l) + \mu_{k,i} - T(t(e_i^j)) \overset{(4.5)}{\le} LC_k(e_k^l) - T(t(e_k^l)) \overset{(4.9)}{\le} e^+$$

und (β) falls beim Maximum in (4.2) der zweite Fall zutrifft:

$$\overset{(4.2,2.\text{Ausdruck})}{=} LC_i(e_i^{j-1}) + \delta_i - T(t(e_i^j)) \overset{(4.4)}{\le} LC_i(e_i^{j-1}) - T(t(e_i^{j-1})) \overset{(4.9)}{\le} e^+$$

und (γ) falls beim Maximum in (4.2) der dritte Fall zutrifft:

$$\overset{(4.2,3.\text{Ausdruck})}{=} C_i(t(e_i^j)) - T(t(e_i^j)) \overset{\text{Prämisse}}{\le} e^+$$

D.h. in allen drei Unterscheidungen ($\alpha - \gamma$) liegt ein Widerspruch zur Annahme (4.8) vor.

Fall b) $LC(e_i^j) = LC_i(e_i^j)$ sei definiert mittels (4.3). Dann gilt:

$$\begin{aligned} LC(e_i^j) - T(t(e_i^j)) &\overset{(4.6)}{=} LC_i(e_i^j) - T(t(e_i^j)) \\ &\overset{(4.3)}{=} \max(LC_i(e_i^{j-1}) + \delta_i,\, C_i(t(e_i^j))) - T(t(e_i^j)) \\ &\overset{\text{Falla})(\beta),(\gamma)}{\le} e^+ \end{aligned}$$

ebenfalls im Widerspruch zur Annahme (4.8).

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

□

Satz 6 *Wenn in einem Prozeß i die Uhr C_i nie mehr als e_i^- nachgeht, dann geht auch die einfache logische Uhr LC in diesem Prozeß nie um mehr als e_i^- nach, d.h.:*

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

Beweis:

$$\begin{aligned} & T(t(e_i^j)) - LC(e_i^j) \overset{(4.6)}{=} T(t(e_i^j) - LC_i(e_i^j)) \\ \overset{(4.2,4.3)}{=} \; & T(t(e_i^j)) - \left\{ \begin{array}{ll} \max(LC_k(e_k^l) + \mu_{k,i},\, LC_i(e_i^{j-1}) + \delta_i,\, C_i(t(e_i^j))) & \text{für ...} \\ \max(LC_i(e_i^{j-1}) + \delta_i,\, C_i(t(e_i^j))) & \text{andernfalls} \end{array} \right\} \\ \overset{-\max(\text{...letzterTerm})}{\le} \; & T(t(e_i^j)) - \left\{ \begin{array}{l} C_i(t(e_i^j)) \\ C_i(t(e_i^j)) \end{array} \right\} \\ \overset{\text{Prämisse}}{\le} \; & e_i^- \end{aligned}$$

□

Bemerkung 5 *Die beiden Sätze 5 und 6 sind nicht* symmetrisch *zueinander: Das Vorgehen der einfachen logischen Uhr 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.*

(Beweis durch entsprechende Wahl von $T := C_i$ bzw. $\max_k C_k$ und $e_i^- := 0$ bzw. $e^+ := 0$.)

4.3 Die geregelte logische Uhr

Im Algorithmus 2 bewirkt der Term $LC_k(e_k^l) + \mu_{k,i}$ in (4.2) ggf. ein Vorstellen der logischen Uhr $LC(e_i^j)$ 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 (4.2) und (4.3), daß LC im Prinzip **stehenbleibt** (bzw. – genau gesagt – um eine kleine Zeitspanne δ_i vorgestellt wird), bis sie dann wieder auf den Wert $C_i(t(e_i^j))$ *zurückgefallen* ist (siehe Term $C_i(t(e_i^j))$ in (4.2) und (4.3)). Damit die logische Uhr gleichmäßiger gehen kann, bzw. um das abwechselnde Vorstellen und Stehenbleiben derselben zu verhindern, wird die einfache logische Uhr aus Algorithmus 2 folgendermaßen erweitert:

Algorithmus 3: LC' bildet die Basis für die geregelte logische Uhr:

$$LC_i'(e_i^j) := \begin{cases} \max(LC_k'(e_k^l) + \mu_{k,i},\ LC_i'(e_i^{j-1}) + \delta_i, \\ \qquad LC_i'(e_i^{j-1}) + \gamma_i^j(C_i(t(e_i^j)) - C_i(t(e_i^{j-1}))), \\ \qquad C_i(t(e_i^j))) \quad \text{wenn } \exists_{e_k^l}\ (e_k^l, e_i^j) \in M & (4.10) \\ \max(LC_i'(e_i^{j-1}) + \delta_i,\ LC_i'(e_i^{j-1}) + \gamma_i^j(C_i(t(e_i^j)) - C_i(t(e_i^{j-1}))), \\ \qquad C_i(t(e_i^j))) \quad \text{andernfalls} & (4.11) \\ \text{wobei die Terme } LC_i(e_i^{j-1}) + \delta_i \text{ bei } j = 0 \text{ entfallen} \end{cases}$$

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

Die zugehörige globale logische Uhr wird wie in Algorithmus 2, Gleichung (4.6), ebenfalls aus den logischen Uhren der einzelnen Prozesse gebildet:

$$LC'(e_i^j) := LC_i'(e_i^j) \tag{4.12}$$

Der Algorithmus wird mit dem Regler in Kap. 4.4 später vervollständigt.

Offensichtlich gilt $LC_{\text{Alg.2}} = LC'_{\text{Alg.3}}$ für $\gamma_i^j \equiv 0$. Für $\gamma_i^j \equiv 1$ entspricht LC' der Definition von Lamport [Lam78] für eine stückweise differenzierbare logische Uhr LC mit $d\,LC_i/dt = d\,C_i/dt$ für alle Zeitpunkte zwischen Empfangsereignissen.

Abb. 4.3 auf Seite 30 zeigt die Wirkungsweise dieses Algorithmus für $\gamma \equiv 1$. Durch die erste Nachricht wird die C_2-Kurve parallel nach oben verschoben (d.h. vorgestellt); das ergibt LC_2' und ist in der Abbildung durch die vertikalen gestrichelten Parallelverschiebungspfeile markiert. Da C_1 die am weitesten vorgehende Uhr ist, wird sie nicht noch weiter vorgestellt, d.h. $LC_1' := C_1$. Damit verlaufen nun LC_1' und LC_2' nahezu gleich, abgesehen von kleinen Schwankungen der Uhrengeschwindigkeiten. Diese Schwankungen können aber dazu führen, daß LC_2' weiter als die schnellste Prozessoruhr (hier C_1) vorgeht, siehe die mit eingekreisten Ereignisse in Abb. 4.3. Dies zu verhindern, ist die Aufgabe des in den nächsten Kapiteln

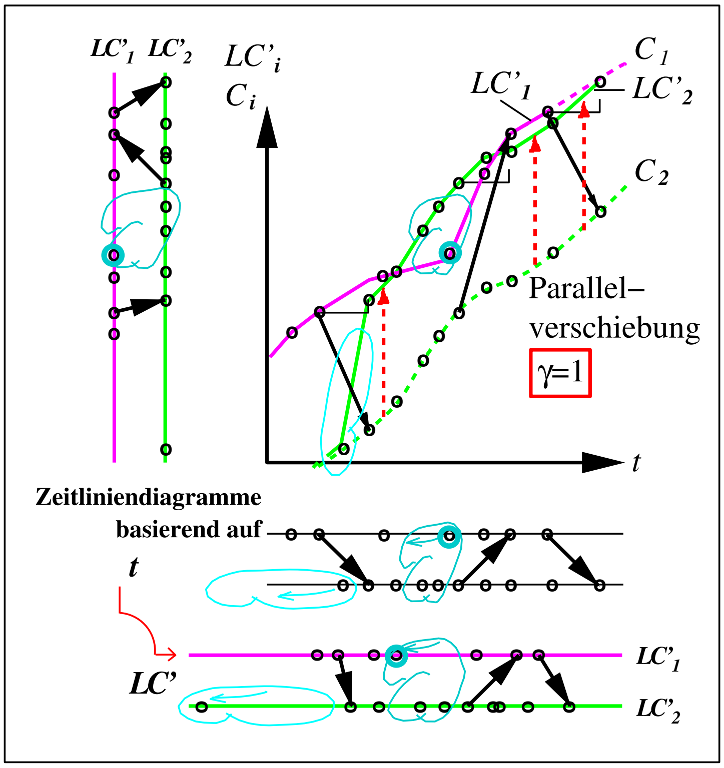


Abb. 4.3: Die geregelte logische Uhr

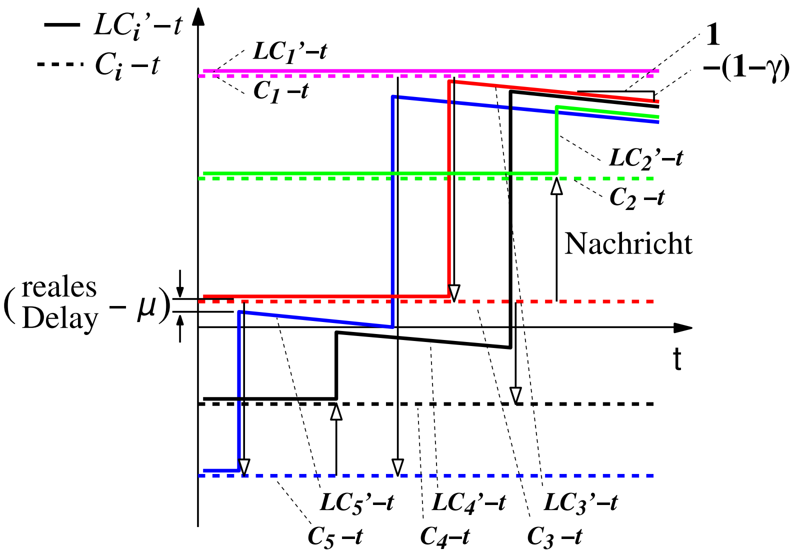


Abb. 4.4: Die geregelte logische Uhr – eine andere Darstellung

beschriebenen Reglers. Außerdem hat man zu Beginn starke Fehler in der Darstellung, solange die Uhrenangleichung mangels entsprechender Nachrichten noch nicht durchgeführt ist. Dies wird deutlich, wenn man das erste Ereignis in Prozeß 2 betrachtet (es ist jeweils mit einer Wolke  eingekreist). Während dieses Ereignis in Wirklichkeit nahe an dem zweiten Ereignis im selben Prozeß liegt (siehe das obere, auf der physikalischen Zeit t basierende Zeitliniendiagramm), wird es in dem auf LC' basierenden, unteren Diagramm lange Zeit vor dem zweiten Ereignis dargestellt. Dies kann am einfachsten dadurch verhindert werden, daß vor dem Start der zu beobachtenden Anwendung sämtliche Prozessoruhren synchronisiert werden. Das untere Zeitliniendiagramm basiert auf der geregelten logischen Uhr und zeigt gegenüber der darüberliegenden Darstellung, die auf der physikalischen globalen Zeit basiert, abgesehen von den soeben beschriebenen Problemen, nur noch geringe Abweichungen in der Darstellung.

Abb. 4.4 auf Seite 30 stellt die Funktionsweise der geregelten logischen Uhr in einer anderen Form dar. Auf der Ordinate sind nicht die Uhren (C, LC), sondern ihr Vorgehen ($C - t$, $LC - t$) aufgetragen. Als Beispiel dienen hier Prozessoruhren, die konstant vor- bzw. nachgehen. Man kann die Wirkungsweise der geregelten logischen Uhr und den Einfluß realer Nachrichtenlaufzeiten und des Faktors γ hier sehen:

- LC'_i wird bei einem Empfangsereignis (bis auf die reale Nachrichtenlaufzeit [delay] abzüglich μ) auf das Niveau der LC'_k des Senders angehoben, wenn dessen Uhr entsprechend stärker vorgeht,
- die angehobenen Kurven $LC'_i - t$ fallen um die Steigung $-(1 - \gamma)$ gegenüber einer gedachten Parallele von $C_i - t$.

Außerdem sieht man, wenn es Nachrichtenketten[4] von dem Prozeß mit der am weitesten vorgehenden Prozessoruhr (hier C_1) zu jedem anderen Prozeß gibt, und deren Nachrichtenlaufzeiten in der Nähe von μ liegen, und wenn $\gamma \cong 1$, daß dann für die geregelten logischen Uhren aller Prozesse

$$LC'_i \cong \max_{k=1..n} (C_k)$$

gilt und somit auch global:

$$LC' \cong \max_{k=1..n} (C_k)$$

Hieraus ergibt sich, daß der Regler den Wert von γ nahe 1 belassen sollte. Am Beispiel in Abb. 4.4 sieht man dies auf der rechten Seite: Nachdem es Nachrichtenketten vom Prozeß 1 zu jedem der anderen vier Prozesse gab, liegen die geregelten logischen Uhren LC'_i nun in der Nähe der Prozessoruhr C_1, welche die am weitesten vorgehende ist. Wenn nun γ nahe bei 1 liegt, dann bleiben die geregelten logischen Uhren über einen langen Zeitraum in der Nähe von C_1. Je näher γ bei 1 liegt, desto weiter können die Zeitpunkte auseinanderliegen, an denen Nachrichtenketten wieder zu einer Synchronisierung führen.

[4] Eine Nachrichtenkette von einem Prozeß P_i zu einem Prozeß P_k sei definiert als eine Folge von Nachrichten für die gilt: (a) Die erste Nachricht wird in Prozeß P_i abgesandt, und (b) jede weitere Nachricht wird erst abgesandt, nachdem die vorhergehende im selben (Zwischen-)Prozeß angekommen war, und (c) die letzte Nachricht wird in Prozeß P_k empfangen.

4.4 Der Regler

4.4.1 Reglerbedarf

Der Regler soll verhindern, daß die geregelte logische Uhr weiter vorgeht, als die am weitesten vorgehende Prozessoruhr. Dies läßt sich dadurch erreichen, daß γ_i^j reduziert wird, sobald bei einem Prozessor die geregelte logische Uhr LC'_i weiter vorgeht, als alle Prozessoruhren, d.h. sobald folgende Bedingung erfüllt ist:

$$\underset{i}{\exists}\, LC'_i > \max_{k=1..n}(C_k)$$

Dies kann vorkommen, wenn

$$\rho_{\text{am weitesten vorgehende Uhr}} < \rho_{\text{beliebige Uhr}} - (1-\gamma) \tag{4.13}$$

mit

$$\rho \;=\; dC/dt - 1 \quad \text{(drift)} \tag{4.14}$$

Abb. 4.5 auf Seite 32 veranschaulicht dies auf der linken Seite.

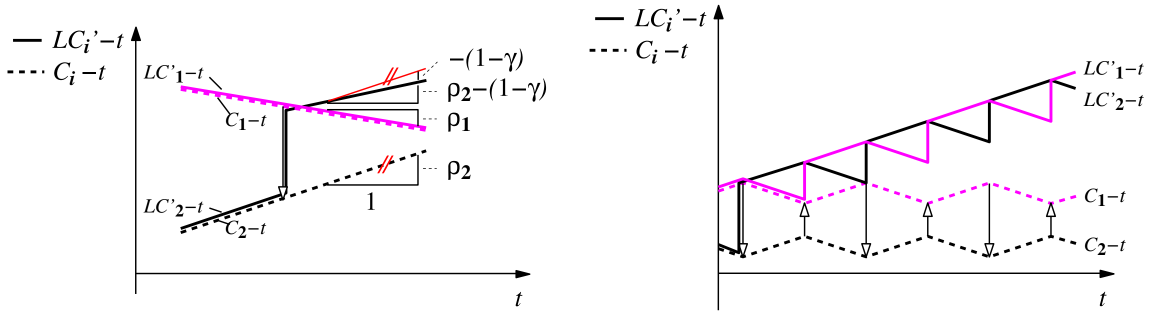


Abb. 4.5: Unbeschränktes Vorgehen der geregelten logischen Uhr bei zu großem γ

C_1 ist die am weitesten vorgehende Prozessoruhr, C_2 eine beliebige andere, deren momentane Gangabweichung so groß (positiv) ist, daß die Ungleichung (4.13) erfüllt ist, d.h. $\rho_2 - \rho_1 > 1-\gamma$ gilt. Solange noch keine Nachricht das Vorstellen von LC'_2 ausgelöst hat, sind die geregelten logischen Uhren LC'_1 und LC'_2 (durchgezogenen Linien, links von dem vertikalen Nachrichtenpfeil) noch identisch mit ihren Prozesoruhren C_1 und C_2 (gestrichelte Linien). Bei dem Nachrichtenaustausch wird nun LC'_2 auf LC'_1 vorgestellt. Danach besitzt LC'_2 die Gangabweichung $\rho_2 - (1-\gamma)$. Da in diesem Beispiel die Ungleichung (4.13) erfüllt ist, geht nun LC'_2 weiter vor als LC'_1.

Während der Satz 6 über die Beschränktheit des Nachgehens ebenfalls für LC' gilt[5], kann man zeigen, daß der Satz 5 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 logische Uhr LC' mehr als e^+ vorgeht. Die rechte Seite von Abb. 4.5 zeigt ein Beispiel. Die gestrichelten Linien geben den Verlauf von C_1 und C_2 an, genauer gesagt von $C_1 - t$ und $C_2 - t$. Die momentane Gangabweichung der Prozessoruhren ist in diesem Beispiel periodisch schwankend. Während die eine Uhr eine

[5]Der Beweis kann direkt übernommen werden

positive Gangabweichung besitzt, hat die andere eine negative. Die Uhrengeschwindigkeiten und damit die Gangabweichungen ändern sich gerade zu dem Zeitpunkt, wenn jeweils die durch die Pfeile symbolisierten Nachrichten ausgetauscht werden. Die durchgezogenen Linien sind die daraus resultierenden geregelten logischen Uhren, aber eben noch ohne Regelung, d.h. mit einem festen γ mit $(1-\gamma)$ kleiner als die absolute Differenz der Gangabweichungen der beiden Prozessoruhren. Die Nachrichten bewirken, daß immer abwechselnd LC'_1 und LC'_2 jeweils auf die andere momentan nachgehende LC'_2 bzw. LC'_1 vorgestellt wird. Folglich wächst der Abstand von LC'_1 und LC'_2 zu ihren Prozessoruhren C_1 und C_2 mit fortschreitender Zeit t immer mehr an. Somit ist das Vorgehen unbeschränkt. Und der alternierende Verlauf der Gangabweichungen der Prozessoruhren sorgt dafür, daß immer die andere logische Uhr nachgeht.

Weitere Beispiele findet man in [Rab96]. Da aber, um die Gangabweichung der geregelten logischen Uhr gering zu halten, γ-Werte im Bereich nahe 1 wünschenswert sind, ist es sinnvoll, $\gamma_i^j \in [0,1]$ über einen Regelkreis zu steuern und soweit herunterzuregeln, daß die Uhren LC'_i nur unbedeutend gegenüber der am weitesten vorgehenden Prozessoruhr vorgehen.

4.4.2 Der Regelkreis

Abb. 4.6 auf Seite 34 zeigt den prinzipiellen Aufbau des Regelkreises der geregelten logischen Uhr. Es handelt sich hierbei um eine diskrete Regelung auf der Ereignismenge $E = \{e_i^j | i = 1..n, j = 0..j_{\max}(i)\}$. Der Index j entspricht der Zeitabszisse des Regelkreises, während der Index i die vektorielle Darstellung aller im Regelkreis beteiligten Größen begründet. Der Regler soll verhindern, daß LC' weiter vorgeht als die am weitesten vorgehende Prozessoruhr $(\max_{k=1..n}(C_k))$, d.h. die Abweichung $\max_{k=1..n}(C_k(t(e_i^j))) - LC'(e_i^j)$ sollte positiv bleiben. Die Führungsgröße des Regelkreises ist daher immer Null. Da aber die Abbildung $t \mapsto C_k(t)$ unbekannt ist, kann dieses Maximum nicht berechnet werden, und somit kann dieses Kriterium nicht direkt erfaßt werden. Daher sind andere Kriterien nötig. Im folgenden werden drei Regelmethoden $\mathcal{A}$, $\mathcal{B}$ und $\mathcal{C}$ und eine Regelbegrenzung $\mathcal{D}$ vorgestellt. Diese werden kombiniert, indem das Minimum der aus den Methoden $\mathcal{A}$, $\mathcal{B}$ und $\mathcal{C}$ resultierenden γ-Werte gebildet wird und dann nach unten mit $\gamma_{\mathcal{D}}$ begrenzt wird:

$$\gamma = \max(\min(\gamma_{\mathcal{A}}, \gamma_{\mathcal{B}}, \gamma_{\mathcal{C}}), \gamma_{\mathcal{D}})$$

Außerdem sind die Meßglieder für die einzelnen Regelmethoden unterschiedlich. Abb. 4.7 auf Seite 34 zeigt den detaillierten Aufbau mit den verschiedenen Regelmethoden. Die dünneren Linien bei den Reglern $\mathcal{A}$, $\mathcal{B}$ und $\mathcal{D}$ zeigen an, daß hier γ eine nicht vektorielle Größe ist, d.h. für alle Prozesse $\{P_i\}_{i=1..n}$ gleich ist. Für vektorielle Größen wurden dickere Linien benutzt. Außerdem ist schon vorweggenommen, daß die Methoden $\mathcal{A}$ und $\mathcal{D}$ Steuerungsglieder sind, d.h. keinen Rückwirkungseingang besitzen. Die Führungsgröße Null wurde zur Vereinfachung weggelassen.

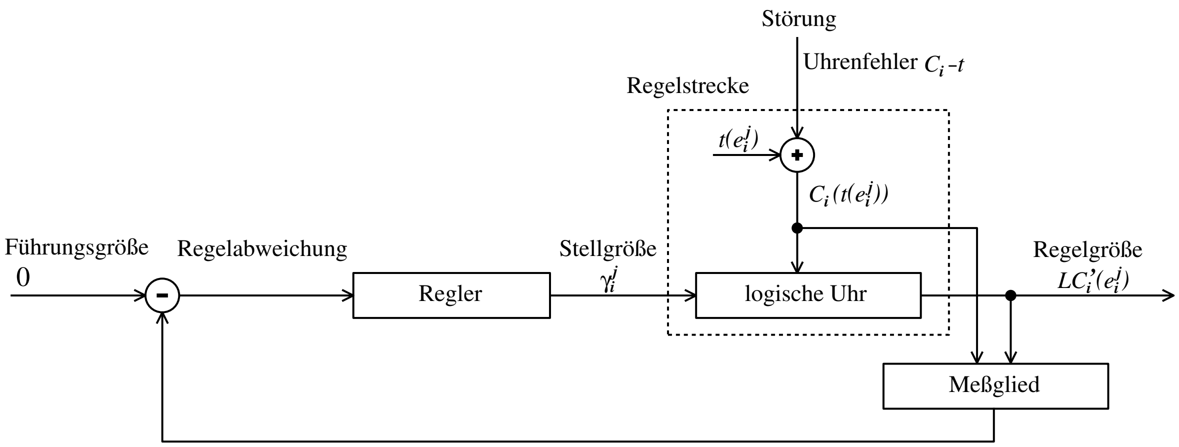


Abb. 4.6: Regelkreis der geregelten logischen Uhr

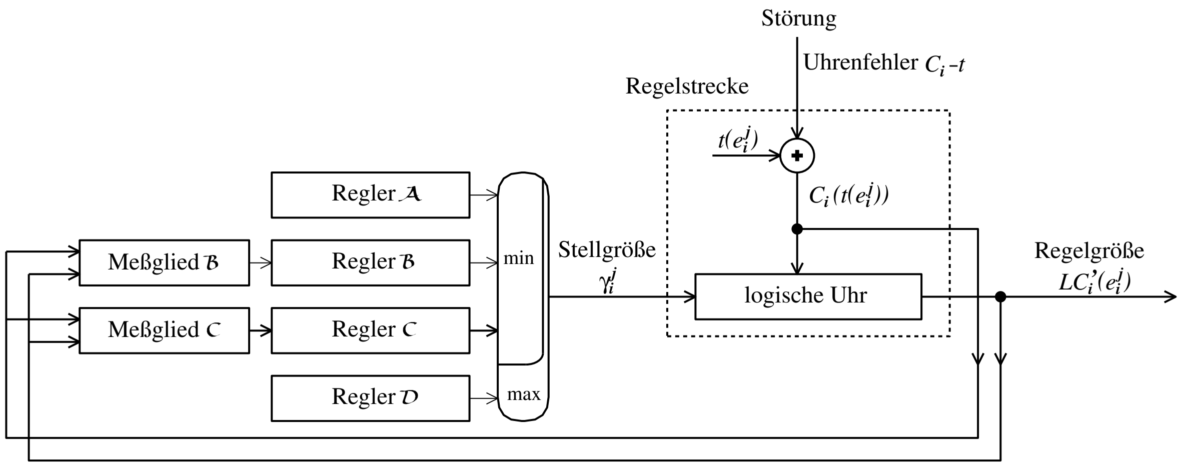


Abb. 4.7: Detaillierte Darstellung des Regelkreises

4.4.3 Regler $\mathcal{A}$ – eine obere Schranke

Satz 7 *Es sei $\gamma_{\max}$ eine Konstante $\in (0,1)$. Wenn alle $\gamma_i^j \leq \gamma_{\max}$ und alle $|\rho| \leq \frac{1-\gamma_{\max}}{2}$ dann bleibt $LC_i' \leq \max_{k=1..n}(C_k)$.*

Beweis: Der Satz folgt direkt aus der Wirkungsweise der geregelten logischen Uhr, dargestellt in Abb. 4.5, links, auf Seite 32. Der Index $i = 1$ repräsentiere bei einer Nachricht den Prozeß, dessen Prozessoruhr C_i am weitesten vorgeht. Es ist im wesentlichen zu zeigen, daß nach dem Vorstellen von LC_2' auf LC_1' $(\leq C_1)$ nun LC_2' nicht gegenüber C_1 aufgrund des Termes $LC_2'(e_2^{j-1}) + \gamma_2^j(C_2(t(e_2^j)) - C_2(t(e_2^{j-1}))$ in Gleichung (4.10) vorgehen kann. Aus der Voraussetzung folgt, daß $-\frac{1-\gamma_{\max}}{2} \leq \rho_1$ und $\rho_2 \leq \frac{1-\gamma_{\max}}{2}$ ist. Es ist also zu zeigen, daß $\rho_2 - (1-\gamma) \leq \rho_1$. Beweis: $\rho_2 - 1 + \gamma \leq \frac{1-\gamma_{\max}}{2} - 1 + \gamma_{\max} = -\frac{1-\gamma_{\max}}{2} \leq \rho_1$

$\square$

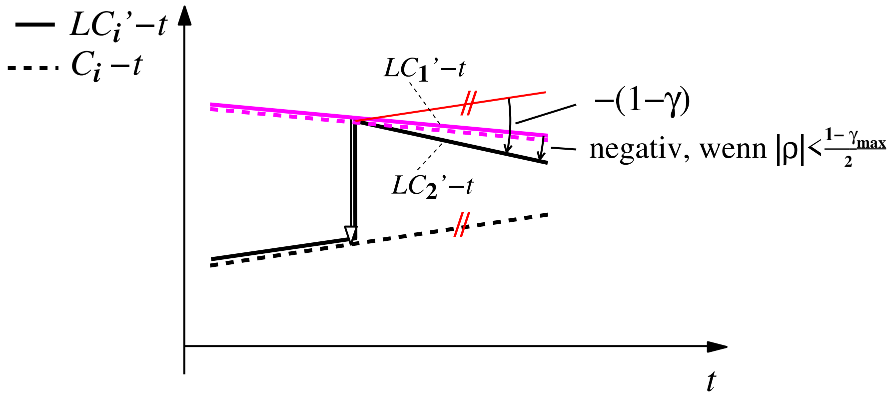


Abb. 4.8: $\gamma_{\max}$, eine Obergrenze für γ

Durch die Wahl einer oberen Grenze $\gamma_{\mathcal{A}} := \gamma_{\max}$ wird das Problem somit gelöst, wenn die Gangabweichungen der Prozessoruhren kleiner als $\frac{1-\gamma_{\max}}{2}$ sind, wie in Abb. 4.8 dargestellt: Es handelt sich hierbei um das gleiche Beispiel wie in Abb. 4.5, nur daß nun $(1-\gamma_{\max})$ so groß gewählt ist, daß LC_2' unter LC_1' bleibt.

In den meisten Fällen ist das Problem des zu starken Vorgehens von LC' damit gelöst. Für z.B. $\gamma_{\max} = 1 - 2 \cdot 10^{-5}$ ist somit das Problem für Gangabweichungen $|\rho| \leq 10^{-5}$ gelöst. Damit sind alle Probleme basierend auf typischen Quarzdrifts im Bereich von 10^{-6} und Gangabweichungen aufgrund schlechter Uhreneinstellungen im Bereich 10^{-5} gelöst. Der hierdurch induzierte Fehler bei Messungen innerhalb eines Prozesses ist auf $1 - \gamma_{\max} = 2 \cdot 10^{-5}$ beschränkt und damit vernachlässigbar. Wenn die Gangabweichung einer Prozessoruhr aber größer ist als $\frac{1-\gamma_{\max}}{2}$, dann ist es möglich, daß die geregelte logische Uhr LC' mehr als die am weitesten vorgehende Prozessoruhr vorgeht. Dieses Problem wird von den beiden folgenden Regelmechanismen gelöst werden.

4.4.4 Regler $\mathcal{B}$ – eine notwendige Regelung

Satz 8 *Es gilt* $\min_k(LC'_k - C_k) > 0 \implies \underset{i}{\exists} LC'_i > \max_{k=1..n}(C_k)$

Beweis: Für i = Index mit $C_i = \max_{k=1..n}(C_k)$ ergibt sich die rechte Seite, da aufgrund der Prämisse $LC'_i > C_i$ gilt.

□

Dieser Satz besagt, daß $\min_k(LC'_k - C_k) > 0$ ein hinreichendes Kriterium dafür ist, daß γ reduziert werden muß, bzw. wenn γ durch dieses Kriterium reduziert wird, dann ist diese Reduktion **notwendig**.

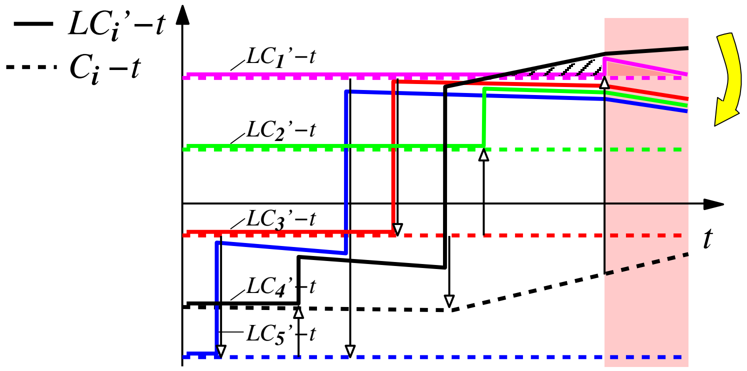


Abb. 4.9: Eine notwendige Regelung

Abb. 4.9 veranschaulicht dieses Kriterium. Das Beispiel ist so konstruiert, daß die Prozessoruhr C_4 eine positive Gangabweichung besitzt. Diese ist damit positiver, als die, der am weitesten vorgehenden Prozessoruhr, hier C_1 mit $\rho = 0$. Durch die aus vierter und fünfter Nachricht gebildeten Nachrichtenkette wird LC'_4 auf den Wert von LC'_1 vorgestellt.[6] Da die Uhrengeschwindigkeit von C_4 größer als die von C_1 ist, geht anschließend LC'_4 gegenüber LC'_1 vor. Dies ist mit dem schräg schraffierten Bereich gekennzeichnet. Die Prämisse von Satz 8 wird nun erfüllt, sobald LC'_1 größer wird als ihre Prozessoruhr. Dies wird durch das Vorstellen von LC'_1 aufgrund der letzten Nachricht von Prozeß 4 nach Prozeß 1 ausgelöst. Nachdem nun auch LC'_1 gegenüber C_1 vorgeht (siehe unterlegtes Dreieck) gehen nun in dem unterlegten Streifen alle logischen Uhren gegenüber ihren Prozessoruhren vor.

Da dieses Kriterium aber nicht aufzeigt, welche geregelten logischen Uhren weiter als erwünscht vorgehen (hier LC'_1 und LC'_4), ist es sinnvoll, global alle γ_i zu reduzieren. Dies führt zu der mit dem breiten, gebogenen Pfeil markierten Absenkung der Uhrengeschwindigkeiten aller LC'_i solange alle weiter als ihre zugehörige Prozessoruhr vorgehen. Für die Reduktion, d.h. für $\gamma_{\mathcal{B}}$ wird die in Abb. 4.10 dargestellte Regler-Charakteristik angewandt.

Die hierbei beim Divisor angewandte Normierung ist von minderer Bedeutung. Daher kann die Maximum-Norm, die bei einer großen Anzahl von Prozessen numerisch nicht skaliert, jederzeit auch durch die Norm mittels der 16. Potenz ersetzt werden. Die Rechenzeit, um die hier vorgesehene Maximum-Norm auszurechnen, ist proportional zur Anzahl n der Prozesse,

[6] Der resultierende Wert kann geringer sein, da die Differenz von realer Nachrichtenlaufzeit minus der minimalen Nachrichtenlaufzeit μ jeweils abgezogen wird, und da der Faktor γ bei einer Nachrichtenkette eine Verlangsamung der dazwischen liegenden Uhren (hier LC'_3) bewirkt.

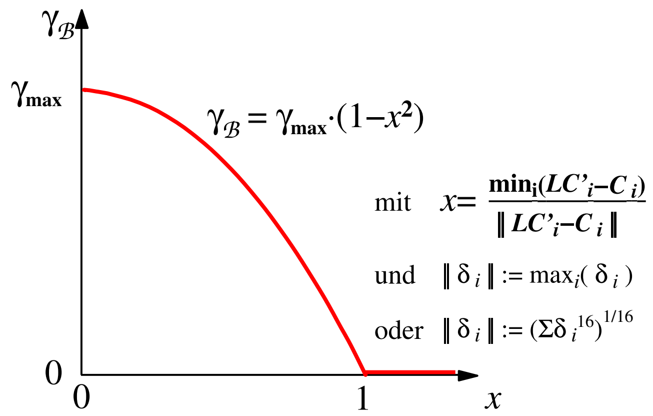


Abb. 4.10: Regler-Charakteristik der notwendigen Regelung

da bei jedem Schritt sich ein $\delta_i := LC'_i - C_i$ ändert, und somit das Maximum über alle n Werte von δ_i wieder neu gebildet werden muß, wenn der alte Wert von δ_i das Maximum lieferte. Ersetzt man hingegen die Maximum-Norm durch die Norm mittels der 16. Potenz, welche numerisch für n Prozesse zwischen der Maximum-Norm und deren $\sqrt[16]{n}$-fachen Wert liegt, d.h. für $n = 1000$ im Bereich des 1 bis 1.54-fachen, dann muß in jedem Schritt der alte Wert von $(\delta_i)^{16}$ von der Summe abgezogen, der neue hinzuaddiert und anschließend die Wurzel berechnet werden. Dieser Aufwand ist zwar pro Iteration groß, aber von der Anzahl der Prozesse unabhängig und für große n geringer als der Aufwand $O(n)$ für die Maximum-Norm.

Das in diesem Absatz angegebene hinreichende Kriterium ist aber nicht notwendig, d.h. es gibt Fälle, in denen eine Reduktion von γ nötig wäre, aber trotzdem $\min_k(LC'_k - C_k) = 0$ ist. Dies ist in dem Beispiel in Abb. 4.9 in dem schräg schraffierten Bereich bei LC'_4 gegeben. Mögliche Szenarien für diesen Fall sind:

- Kurze Phasen bevor durch eine Nachricht der Regler ausgelöst wird, wie in obigem Beispiel.
- Es gibt Gruppen von Prozesse, die zwar innerhalb der Gruppe kommunizieren, aber über längere Zeit nicht zwischen den Gruppen. Dann kann innerhalb einer Gruppe eine logische Uhr übermäßig vorgehen, obwohl das Kriterium nicht anspricht, da in einer anderen Gruppe mindestens eine logische Uhr nicht vorgeht, d.h. $LC'_i = C_i$ ist. Dies kann z.B. in Clustern von Parallelrechnern und darauf abgestimmter Anwendungssoftware passieren, vorausgesetzt die Parallelrechner besitzen intern keine Hardwaresynchronisation.

Nur das letztere Szenarium führt in der Regel zu Abweichungen, die einen weiteren Regelmechanismus erforderlich machen. Dieser wird im folgenden Abschnitt beschrieben.

4.4.5 Regler $\mathcal{C}$ – eine hinreichende Regelung

Der dritte Regler sei anhand der Abb. 4.11 erklärt.

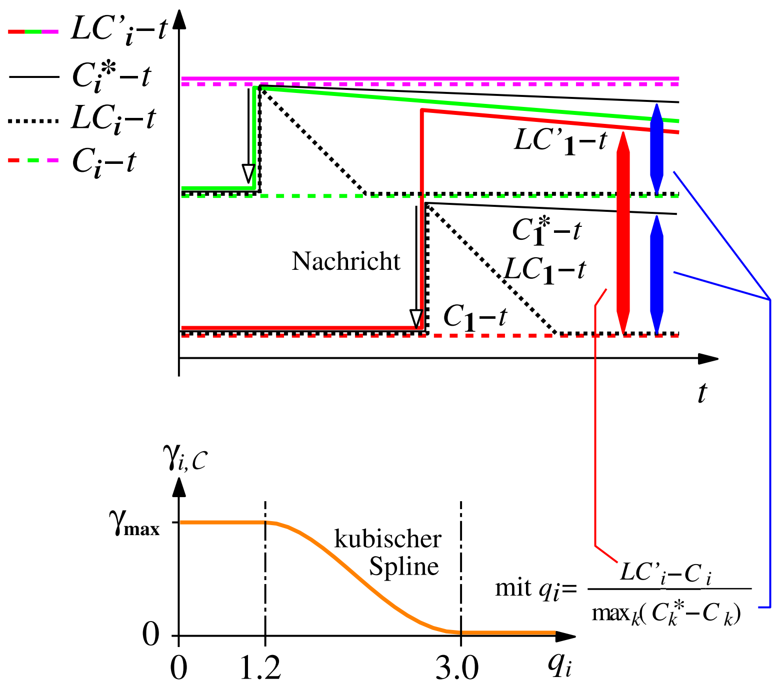


Abb. 4.11: Eine hinreichende Regelung

Zuerst sei die Kurve C_i^* definiert: Sie ist am Anfang identisch mit der einfachen logischen Uhr LC_i. Sie wird immer soweit vorgestellt, daß sie mindestens genauso weit vorgeht wie LC_i, und sie geht mit $[1-\frac{1}{2}(1-\gamma)]dC_i/dt$. Sie ist mit den schwarzen Linien dargestellt. Insgesamt ist aber für den dritten Regler nur $\max_i(C_i^* - C_i)$ relevant. Daher wird folgender Algorithmus angewandt, der im Term $-\frac{1}{2}(1-\gamma)dC_i/dt$ eine geringfügige und vernachlässigbare Abweichung besitzt.

Algorithmus 4:

Variablen:

MAX für $\max_i(C_i^* - C_i)$ für das letzte berechnete Ereignis e_i^j
LC_{MAX} für den den Wert von LC_i bei der letzten Erhöhung von MAX

Initialisierung:

$MAX = 0$ und $LC_{MAX} = LC_i(e_i^0)$ beim ersten Ereignis

Vorschrift bei jedem weiteren Ereignis e_i^j:

$MAX := MAX - \frac{1}{2}(1-\gamma_{\max})max(0, LC_i(e_i^j) - LC_{MAX})$
$MAX := \max(0, MAX)$
if $(LC_i(e_i^j) > C_i(e_i^j))$ then
$MAX = LC_i(e_i^j) - C_i(e_i^j)$
$LC_{MAX} = LC_i(e_i^j)$
endif

Für die Kurven C_i^* gilt, daß ihre Abstände zu C_i genauso beschränkt sind, wie bei den einfachen logischen Uhren LC_i. Aber C_i^* erfüllt nicht die Uhrenbedingung. Im Prinzip handelt es sich bei C_i^* um Parallelverschiebungen wie bei der geregelten logischen Uhr, aber sie kann durch einzelne Nachrichten nur auf die Prozessorzeit des jeweiligen Senders vorgestellt werden, während die geregelte logische Uhr über Nachrichtenketten auf die Prozessorzeit der Initiatoren der Ketten vorgestellt werden kann. Für die Regelung wird nun das Verhältnis

$$\frac{LC_i' - C_i}{\max_k(C_k^* - C_k)}$$

berechnet. Wenn es von dem Prozeß mit der am weitesten vorgehenden Prozessoruhr Nachrichten mit minimaler Laufzeit zu dem Prozeß mit der am weitesten nachgehenden Uhr gibt, dann gibt der Divisor $\max_k(C_k^* - C_k)$ den maximalen Abstand zweier Uhren an. Dieser Regler regelt spezifisch das $\gamma_{i,\mathcal{C}}$ jedes einzelnen Prozesses i auf der Basis der in Abb. 4.11 angegebenen Reglercharakteristik. Diese bewirkt, daß im ungünstigsten Fall der Regler erst das γ_i zu reduzieren beginnt, wenn die geregelte logische Uhr LC_1' um das 1.2-fache der maximalen Uhrendifferenz gegenüber der am weitesten vorgehenden Prozessoruhr vorgeht, d.h. der Regler $\mathcal{C}$ setzt spätestens dann ein, wenn für ein LC_i' folgende Schranke überschritten ist:

$$LC_i' \; > \; \max_k(C_k) + 1.2 \cdot \max_{k,m}(C_k - C_m)$$

Dieser Regler garantiert auf der Basis obiger Überlegung des ungünstigsten Falles und aufgrund der auf Null absinkenden Reglercharakteristik, daß die Differenz $LC' - \max_k(C_k)$ nach oben beschränkt bleibt.

Andererseits hat der Regler absichtlich eine Regelkurve, die vom 1.2-fachen bis zum 3-fachen geht, da das Kriterium dieses Reglers kein hinreichendes ist, d.h. es gibt Fälle in denen der Regler γ herunterregelt, obwohl es dafür keine Notwendigkeit gibt, also alle LC_i' kleiner oder gleich $\max_k(C_k)$ sind. Der Regler kann zu restriktiv sein, wenn

$$\max_{\substack{i,k \text{ mit Nachrichten} \\ \text{von } i \text{ nach } k}}(C_i - C_k) < 1.2 \cdot \max_{i,k}(C_i - C_k)$$

wobei der Faktor 1.2 in der Praxis aufgrund des Unterschieds zwischen den real auftretenden Nachrichtenlaufzeiten und der minimalen Laufzeit meist größer ist. Der Fall eines zu restriktiven Reglers kann z.B. auftreten, wenn bei einer matrixförmigen Anordnung der Prozessoren die Uhren von einer Seite zur anderen immer mehr nachgehen, und die Anwendung nur Nachrichten zwischen den direkten Nachbarn austauscht.

4.4.6 Regler $\mathcal{D}$ – zur Fehler-Begrenzung

Die Regler $\mathcal{B}$ und $\mathcal{C}$ können γ im Extremfall auf Null herab regeln. Dies würde bedeuten, daß die betroffene geregelte logische Uhr für einige Zeit nahezu stehen bliebe. Um das Auseinanderdriften von C und LC' zu verhindern, genügt es aber, daß das γ im Extremfall die schnellste Prozessoruhr auf die Geschwindigkeit der langsamsten Prozessoruhr regelt.

$\gamma_{\mathcal{D}}$ sei daher eine Konstante mit $\gamma_{\mathcal{D}} < 1 - 2\max|\rho|$. Da $\gamma_{\mathcal{D}}$ den Fehler bzgl. des zu langsam Gehens der geregelten logischen Uhren begrenzt und in der Praxis normalerweise nicht mit einem $|\rho| > 1\%$ zu rechnen ist, sei der Wert $\gamma_{\mathcal{D}} = 0.98$ empfohlen. In Systemen, bei denen die Uhrenfehler nur auf den unterschiedlichen Gangabweichungen der Uhrenquarze beruhen, kann man $|\rho| < 5 \cdot 10^{-5}$ annehmen und $\gamma_{\mathcal{D}} = 10^{-4}$ wählen.

4.4.7 Zusammenfassung der Regler

Die Regler werden mittels $\gamma_i^j = \max(\min(\gamma_{\mathcal{A}}, \gamma_{\mathcal{B}}, \gamma_{i,\mathcal{C}}), \gamma_{\mathcal{D}})$ kombiniert. Der Regler $\mathcal{A}$ ist ausreichend, wenn die Gangabweichungen der Prozessoruhren hinreichend klein sind. Dieser Regler ist somit in der Praxis der am meisten relevante. Es wird $\gamma_{\max} = 1-2\cdot 10^{-5}$ empfohlen. Der Regler $\mathcal{B}$ ist in den meisten anderen Fällen ausreichend, aber nicht in allen. Er hat ein minimales Überschwingen und kontrolliert das Ziel ($LC' \cong \max_k(C_k)$) sehr scharf. Der Regler $\mathcal{C}$ ist in allen Fällen ausreichend, aber in manchen zu restriktiv und wurde daher nur mit einer schwach regelnden Charakteristik versehen. Er ist der einzige der γ prozeßspezifisch regelt, d.h. $\gamma_{i,\mathcal{C}}$ kann für unterschiedliche i verschieden sein. Der Regler $\mathcal{D}$ begrenzt γ_i^j nach unten. Da die Charakteristiken in den Reglern $\mathcal{B}$ und $\mathcal{C}$ ebenfalls γ auf maximal $\gamma_{\max}$ beschränken, bedarf der Regler $\mathcal{A}$ keines gesonderten Implementierungsaufwands.

Ein mögliches Oszillieren des Reglers muß hierbei nicht berücksichtigt, bzw. gedämpft, werden, da in der Fehlerbetrachtung im nachfolgenden Kapitel 5 gezeigt wird, daß die Fehler in der Regel kleiner als 5 % sind und damit ein Oszillieren ebenfalls in dieser Fehlertoleranz liegen muß und damit vernachläßigt werden kann.

4.5 Rückwärtige Amortisation

Bei der geregelten logischen Uhr (Alg. 3) können bei Empfangsereignissen durch das Vorstellen der logischen Uhr auf den Zeitstempel der logischen Uhr des zugehörigen Sendeereignisses (Term $LC'_k(e_k^l) + \mu_{k,i}$ in (4.10)) Unstetigkeiten entstehen.

Durch die *rückwärtige Amortisation* werden diese Unstetigkeiten stückweise linear auf ein mit dem Empfangsereignis endendes Zeitintervall verteilt, d.h. diese Sprünge der geregelten logischen Uhr werden beseitigt.

Algorithmus 5: Die rückwärtige Amortisation wird in jedem Prozeß für jedes Empfangsereignis berechnet, bei dem die geregelte logische Uhr durch den oben genannten Term vorgestellt wird. Sie wird in der Reihenfolge der Ereignisse durchgeführt, wobei die einzelnen Prozesse unabhängig voneinander bearbeitet werden. Das Verfahren ist in Abb. 4.12 auf Seite 41 für das rechte Empfangsereignis **R** in Prozeß P_i skizziert. Für frühere Empfangsereignisse ist die Amortisation in dieser Skizze also schon durchgeführt.

LC_i^b ist eine Uhr, die nur bis zum zu bearbeitenden Empfangsereignis **R** im Prozeß i gültig ist, und die folgendermaßen definiert ist: Sie ergibt sich aus der geregelten logischen Uhr zuzüglich rückwärtiger Amortisation vorangegangener Empfangsereignisse, aber abzüglich des bei **R** durch oben genannten Term erzeugten Sprungs. LC_i^b wird bei jeder Amortisation wieder neu gebildet. LC_i^b dient in der Abbildung als Abszisse.

LC_i^{bs} ist wie LC_i^b, aber inklusive des Sprungs bei **R**. $LC_i^{bs} - LC_i^b$ ist in der Abbildung als durchgezogene Linie dargestellt. Sie ist konstant null, bis auf den Sprung bei **R**.

LC_i^A ist die durch diesen Algorithmus definierte *rückwärtig amortisierte geregelte logische Uhr*. $LC_i^A - LC_i^b$ ist in der Abbildung strich-punktiert eingezeichnet.

A_{def} ist der Amortisationsfaktor. Er wird vom Anwender gleich dem von ihm gewünschten Amortisationsfehler gesetzt (z.B. $A_{\text{def}} := 0.02 = 2\,\%$).

Die Länge des Amortisationsintervalls wird bestimmt durch den Quotienten aus der maximal ermittelten Uhrendifferenz und dem Amortisationsfaktor A_{def}. Es endet bei **R**.

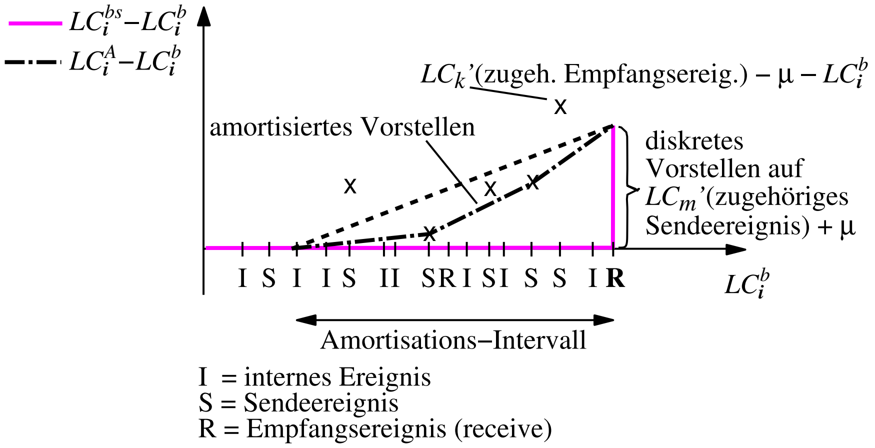


Abb. 4.12: Nachträgliche stückweise lineare Amortisation

Zu Beginn ist vom Anwender eine zu erwartende maximale Uhrendifferenz vorzugeben. Diese wird während der Berechnung der Amortisation erhöht, wenn eine Sprunghöhe über diesem Wert liegt.

Die durchgezogene Linie zeigt den Verlauf der geregelten logischen Uhr LC_i' eines Prozesses i mit einem Empfangsereignis **R**, bei dem die logische Uhr aufgrund des zugehörigen Sendeereignisses vorgestellt werden muß. Hierbei ist auf der Ordinate nicht LC_i' sondern $LC_i' - LC_i^b$ abgetragen, d.h. sie hat vor **R** den Wert Null und an R den Sprung, der durch die rückwärtige Amortisation ausgeglichen werden soll.

Die Kreuze (x) markieren innerhalb des Amortisationsintervalls zu jedem Sendeereignis des Prozesses i den Wert $LC_k'(\text{zugehöriges Empfangsereignis}) - \mu - LC_i^b$. Die Uhrenbedingung garantiert, daß diese Werte immer positiv sind.

Die gestrichelte Gerade verbindet den Wert Null am Anfang des Amortisationsintervalls mit der Sprunghöhe $= LC_i^{bs} - LC_i^b$ des Empfangsereignisses **R**. Falls es kein Ereignis vor dem Amortisationsintervall innerhalb des Prozesses i gibt, beginnt die gestrichelte Gerade beim ersten Ereignis mit dem Wert des untersten Kreuzes, und falls zudem das Amortisationsintervall keine Sendeereignisse enthält, ist die gestrichelte Gerade konstant auf der Sprunghöhe.

Wenn sich kein Kreuz unterhalb der gestrichelten Linie befindet, dann ergibt die gestrichelte Linie den rückwärtig amortisierten Wert LC_i^A der geregelten logischen Uhr in dem Amortisationsintervall. Andernfalls ist der Wert durch die untere konvexe Hüllkurve unterhalb der Kreuze zwischen den Endpunkten der gestrichelten Linie gegeben. In der Skizze ist sie strich-punktiert dargestellt.

Der Amortisationsgradient A bezeichnet die Steigung der konvexen Hülle im jeweiligen Teilintervall.

Im Gegensatz zu der von F. Schmuck und F. Cristian in [SC90] untersuchten Amortisation wird hier die diskrete Änderung nicht in einem an dem Synchronisationszeitpunkt beginnenden Zeitintervall ausgeglichen, sondern der Ausgleich wird rückwirkend in einem Intervall vor dem Synchronisationszeitpunkt durchgeführt. Die zeitliche Länge dieses Amortisationsintervalls ist durch die maximale Uhrendifferenz und die gewünschte Mindestgenauigkeit beschränkt, typischerweise in einem Workstation-Cluster auf z.B. $5\,\mathrm{ms}\,/1\,\% = 0.5\,\mathrm{sec}$. Eine rückwirkende Amortisation ist nur möglich, wenn der Einsatz der geregelten logischen Uhr eine derartige nachträgliche Änderung der Zeitstempel zuläßt. Bei der in Kap. 6 vorgestellten Filterimplementierung ist das möglich, bewirkt aber eine Vergrößerung der zu sortierenden Ereignismenge auf (*die Länge des Amortisationsintervalls*) $\cdot (\Delta E/\Delta t)$. Bei der in Kap. 7 skizzierten online Implementierung im Rahmen eines Debuggingwerkzeuges muß berücksichtigt werden, daß sich die Darstellung durch die Amortisation nachträglich ändert oder erst verspätet zur Verfügung steht. Die Verspätung ist einerseits durch die Länge des Amortisationsintervalls gegeben und andererseits dadurch, daß die Amortisation erst berechnet werden kann, wenn zu jedem Sendeereignis innerhalb des Amortisationsintervalls das zugehörige Empfangsereignis schon stattgefunden hat. Eine nachträgliche Änderung der Darstellung kann, z.B. bei einem Zeitliniendiagramm sukzessive im Rahmen des beim zeitlichen Fortschreiten notwendigen Scrollens realisiert werden, wobei dies für den Anwender in den meisten Fällen verborgen bleibt.

Es bleibt nun zu zeigen, daß die rückwärtig amortisierte geregelte logische Uhr LC^A die Uhrenbedingung (Def. 7) erfüllt:

Satz 9 *Die durch Algorithmus 5 definierte rückwärtig amortisierte geregelte logische Uhr LC^A erfüllt die Uhrenbedingung.*

Beweis: Lamport zeigte in [Lam78], daß es genügt, die Gültigkeit der Regeln IR1 und IR2 – siehe Fußnoten (2) und (3) auf Seite 27 – zu zeigen.

Für zwei aufeinanderfolgende Ereignisse innerhalb des selben Prozesses i gilt: Die Regel IR1 ist für LC_i^b erfüllt, da sie auch schon ohne den Term $LC_k'(e_k^l)+\mu_{k,i}$ in (4.10) erfüllt ist. Da die für die rückwärtige Amortisation addierten Werte mit der Ereignisfolge monoton steigend sind, ist auch für LC_i^A die Regel IR1 erfüllt. Daß die zu addierenden Werte monoton steigend sind, ergibt sich daraus,
(i) daß kein Kreuz unterhalb des Anfangswertes liegt und, daß der Anfangswert nicht kleiner oder gleich dem Endwert (Sprunghöhe) ist, d.h. die Steigung des ersten linearen Abschnitts ist immer größer oder gleich Null und,
(ii) daß die untere **konvexe** Hülle gebildet wurde, d.h. die Steigungen der linearen Abschnitte sind von Abschnitt zu Abschnitt zunehmend.

Für jede beliebige Nachricht ist die Regel IR2 zu zeigen: Die Bildung der **unteren** konvexen Hülle garantiert, daß der beim Sendeereignis addierte Wert immer kleiner oder gleich dem Wert des zugehörigen Kreuzes ist. Dies ergibt sich aus

$$\begin{aligned} LC_i^A(\text{Sendeereignis}) &\overset{(1)}{\leq} LC_i^b + (LC_k'(\text{zugehöriges Empfangsereignis}) - \mu - LC_i^b) \\ &= LC_k'(\text{zugehöriges Empfangsereignis}) - \mu \\ &\overset{(2)}{\leq} LC_k^A(\text{zugehöriges Empfangsereignis}) - \mu \\ &\overset{(3)}{<} LC_k^A(\text{zugehöriges Empfangsereignis}) \end{aligned}$$

Die Abschätzungen gelten
$^{(1)}$ wegen der Definition der Kurve in Abb. 4.12,
$^{(2)}$ da die Amortisation die Zeitstempel nur vergrößert, und
$^{(3)}$ da $\mu > 0$.

□

Definition 9: Im folgenden wird mit LC_i^A, bzw. LC^A die durch Algorithmus 5 beim letzten Empfangs-Ergeignis ermittelte Uhr bezeichnet. Für Sendeereignisse und interne Ergeignisse, die dem letzten Empfangsergeignis nachfolgen, ist LC_i^A identisch mit der geregelten logischen Uhr definiert, d.h. $LC_i^A(e_i^j) := LC_i'(e_i^j)$ für e_i^j nach letztem Empfangsereignis in Prozeß i.

4.6 Zusammenfassung

Die geregelte logische Uhr mit rückwärtiger Amortisation besteht also aus folgenden Komponenten:

- die logische Uhr, die dafür sorgt, daß die logische Uhrenbedingung erfüllt wird,
- die Übernahme der Uhrengeschwindigkeit der Prozessoruhren auf die logische Uhr, damit die Gangabweichungen der logischen Uhr solange gering sind, wie nicht die logische Uhr ein Vorstellen der Zeit durchführt,
- die rückwärtige Amortisation, die gerade dieses Vorstellen der Uhr stückweise linear in der Vergangenheit ausgleicht und damit auch in diesem Fall die Gangungenauigkeit minimal läßt, und
- der Regler, der dafür sorgt, daß die absoluten Synchronisationsfehler sich nicht addieren, sondern daß die logische Uhr möglichst nicht weiter vorgeht als die am weitesten vorgehende Prozessoruhr.

Kapitel 5

Fehleranalyse

5.1 Einführung

Man hat drei Grundtypen von Meßintervallen zu unterscheiden:

d_M Nachrichtenlaufzeit,

d_R Zeitintervall zwischen zwei aufeinanderfolgenden Ereignissen im gleichen Prozeß und das letztere ist ein Empfangsereignis und

$d_{S/I}$ Zeitintervall zwischen zwei aufeinanderfolgenden Ereignissen im gleichen Prozeß, und das letztere ist ein Sendeereignis oder ein internes Ereignis.

Bei dieser Unterscheidung wurde der Funktionsweise der geregelten logischen Uhr Rechnung getragen. Jedes Zeitintervall zwischen zwei beliebigen Ereignissen läßt sich als Summe und Differenz derartiger Intervalle berechnen, vorausgesetzt der Graph aller Nachrichten (Kanten) zwischen den Prozessen (Knoten) ist zusammenhängend.

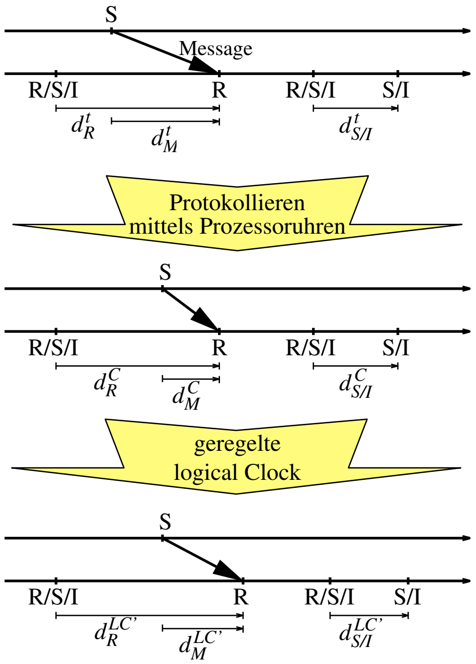


Abb. 5.1: Zeitintervalle (R=Empfangs-, S=Send-, I=Internes Ereignis)

Außerdem ist zu unterscheiden zwischen

d^t exakt gemessenes Zeitintervall,

d^C Zeitintervall gemessen auf der Basis der Prozessoruhren, und

$d^{LC'}$ Zeitintervall bestimmt nach der Korrektur von d^C durch die geregelte logische Uhr.

d^{LCA} Zeitintervall nach der Durchführung der rückwärtigen Amortisation.

Absolute und relative Fehler werden mit e und ϵ bezeichnet, d.h.

$$e^{LC'} = |d^{LC'} - d^t| \tag{5.1}$$

$$\epsilon^{LC'} = |\frac{d^{LC'} - d^t}{d^t}| \tag{5.2}$$

und analog für e^{LCA} und ϵ^{LCA}.
Folgende Abschätzungen sind möglich:

Zu $\mathbf{d_M}$:

Unter der Voraussetzung, daß μ die minimale Nachrichtenlaufzeit ist (d.h. $\forall d^t_M : d^t_M \geq \mu$), gilt $d^{LC'}_M \geq \mu$, d.h. fehlerhafte Messwerte d^C_M mit $d^C_M < \mu$ werden garantiert auf einen Wert $d^{LC'}_M \geq \mu$, bzw. $d^{LCA}_M \geq \mu$ korrigiert. Wenn außerdem zwischen allen Prozessen hinreichend oft Nachrichtenketten existieren, bei denen die einzelnen Nachrichtenlaufzeiten nahe μ sind, dann kann die geregelte logische Uhr in allen Prozessen das Maximum aller Prozessoruhren annähern und damit eine einheitliche globale Zeit bilden und die Fehler $|d^{LC'}_M - d^t_M|$, bzw. $|d^{LCA}_M - d^t_M|$ minimieren. Die durch diese Abschätzungen charakterisierte Korrektur der Nachrichtenlaufzeiten ist ein wesentlicher **positiver Effekt** der geregelten logischen Uhr. Die Beispiele in Kapitel 6.3.1 und 6.3.2 zeigen diese Wirkung insbesondere in der Praxis in den Abb. 6.3 (S. 72) und 6.4 (S. 74).

Die beiden folgenden Abschnitte befassen sich dagegen mit **zusätzlichen Fehlern**, die durch die geregelte logische Uhr entstehen und damit mit deren Nachteilen.

Zu $\mathbf{d_{S/I}}$:

Es sei $\rho_{\max}$ eine obere Grenze für die absolute Gangabweichung der Prozessoruhren. Dann gilt $d^t_{S/I}(1 - \rho_{\max}) \leq d^C_{S/I} \leq d^t_{S/I}(1 + \rho_{\max})$. Aufgrund der Terme mit γ in Alg. 3 gilt für die geregelte logische Uhr $d^C_{S/I}(1 - (1 - \gamma)) \leq d^{LC'}_{S/I} \leq d^C_{S/I}$. Zusammenfassend ergibt sich unter Vernachlässigung höherer Terme

$$d^t_{S/I}(1 - \rho_{\max} - (1 - \gamma)) \leq d^{LC'}_{S/I} \leq d^t_{S/I}(1 + \rho_{\max})$$

und für den relativen Fehler

$$\epsilon^{LC'}_{S/I} := \left| \frac{d^{LC'}_{S/I} - d^t_{S/I}}{d^t_{S/I}} \right| \leq \rho_{\max} + (1 - \gamma)$$

Typischerweise liegt $(1 - \gamma)$ bei $1 - \gamma_{\max} = 2 \cdot 10^{-5}$ und $\rho_{\max}$ bei 10^{-6} und für $\rho_{\max} > 10^{-5}$ sollte der Regler im Normalfall γ nur soweit herunterregeln, daß $1 - \gamma \leq 4\rho_{\max}$ erhalten bleibt (dies berücksichtigt 2-faches Überschwingen und gegenläufige maximale Gangabweichungen, s. auch linkes Diagramm in Abb. 4.5). Damit gilt in der Regel

$$\epsilon^{LC'}_{S/I} \leq \max(2 \cdot 10^{-5} + \rho_{\max}, 4\rho_{\max})$$

d.h. **diese zusätzlichen Fehler** durch die geregelte logische Uhr sind **vernachlässigbar** oder zumindest im Bereich der Ganggenauigkeit der Prozessorclocks. Die rückwärtige Amortisation erzeugt noch geringe zusätzliche additive Fehler. Eine exakte Analyse wird in den nachfolgenden Kapiteln durchgeführt.

Zu $\mathbf{d_R}$:

Aufgrund des möglichen Vorstellens der geregelten logischen Uhr bei Empfangsereignissen (siehe Term $LC'_{k=\mathrm{sender}}+\mu$ in (4.10)) um die Uhrenbedingung (2.4) zu gewährleisten, entstehen bei $d_R^{LC'}$ ggf. größere Fehler. Hierbei sind zwei Phasen zu unterscheiden:

1. Phase: Aufgrund anfänglicher Nachrichten werden die LC_i dem $\max_k(C_k)$ angenähert;

2. Phase: Nötige Korrekturen aufgrund der Gangabweichungen der Prozessoruhren oder wegen späterer Nachrichten, deren Laufzeit näher an μ liegt.

In der 1. Phase sind die absoluten Fehler $e_R := \left|d_R^{LC'} - d_R^t\right|$ im Bereich der Uhrenfehler $|C_i - C_k|$. Dies kann aber durch eine Synchronisation der Uhren vor Beginn der zu beobachtenden Anwendung vermieden werden, da dann zu Beginn $\left|d_R^{LC'} - d_R^t\right| \cong 0$ gilt. In der 2. Phase hängen die Fehler davon ab, ob die Uhren kontinuierlich arbeiten oder ob sie Sprünge machen. Außerdem ist zu berücksichtigen, wie viel Zeit verstreicht, bis die geregelte logische Uhr durch eine geeignete Nachricht korrigiert wird, denn dies kann zu einem größeren Korrekturbedarf und damit zu einem größeren Fehler e_R führen. Die geregelte logische Uhr kann den Fehler e_R für jeden Tracefile und jedes Empfangsereignis berechnen und z.B. Maximum und Durchschnitt ausgeben. Bezüglich des relativen Fehlers $\epsilon_R := e_R/d_R^t$ kann man keine sinnvolle Abschätzung garantieren, da er bei sehr kurzen Intervallen d_R^t beliebig groß werden kann. Doch in der Praxis sollte ϵ_R **im Durchschnitt über alle Empfangsereignisse höchstens einige Prozent** betragen. Kap. 6.3.1 enthält hierzu ein Beispiel. Mit der in Kap. 4.5 vorgestellten nachträglichen stückweise linearen Amortisation werden diese Fehler in der Regel nochmals stark reduziert. Die nachfolgenden Kapitel geben eine präzise Abschätzung für die lokalen Fehler, d.h.für $\epsilon_{R/S/I}$.

Bei Tracefiles bei denen alle Nachrichtenlaufzeiten größer oder gleich μ sind, ist zur Erreichung der Uhrenbedingung keine Korrektur nötig, und es wird auch keine durchgeführt. In diesem Fall entstehen durch die geregelte logische Uhr keine zusätzlichen Fehler.

Zusammenfassend läßt sich sagen, daß in der Regel der Gewinn durch die sinnvolle Korrektur der Nachrichtenlaufzeiten (d_M) die zusätzlichen Fehler bei Zeitintervallen innerhalb eines Prozesses ($d_{S/I}$ und d_R) bei weitem übersteigt. Durch eine präzisere Bestimmung von μ für jede einzelne Nachricht in Abhängigkeit von deren Länge ließe sich das Verfahren bei Bedarf noch deutlich verbessern.

5.2 Charakteristik des Reglers $\mathcal{A}$

Der Regler $\mathcal{A}$ soll anhand des folgenden Szenariums analysiert werden.

Szenarium 1 *Die Uhr von Prozeß 0 sei exakt, d.h.* $C_0(t) = t$. *Die Uhr von Prozeß 1 gehe (zum Zeitpunkt* $t = 0$*) um* D *nach und sie geht um* ρ *schneller als* C_0*, d.h.* $C_1(t) = t(1+\rho)-D$. *Es sei* $\gamma_{\max}(1+\rho) \leq 1$. *Damit ist die geregelte logische Uhr* LC'_0 *des Prozeß 0 immer identisch mit* C_0 *(solange* $C_1 < C_0$*). Das Ereignis* e_1^j *in Prozeß 1 ist ein Empfang einer Nachricht von Prozeß 0.* $\tilde{\tau}$ *nach* e_1^j *gibt es in Prozeß 1 ein weiteres Ereignis* e_1^{j+1}. *Und* τ *nach* e_1^j *gibt es in Prozeß 1 das Ereignis* e_1^{j+2}*, das nach* e_1^j *das nächste Empfangen einer Nachricht von Prozeß 0 ist.*

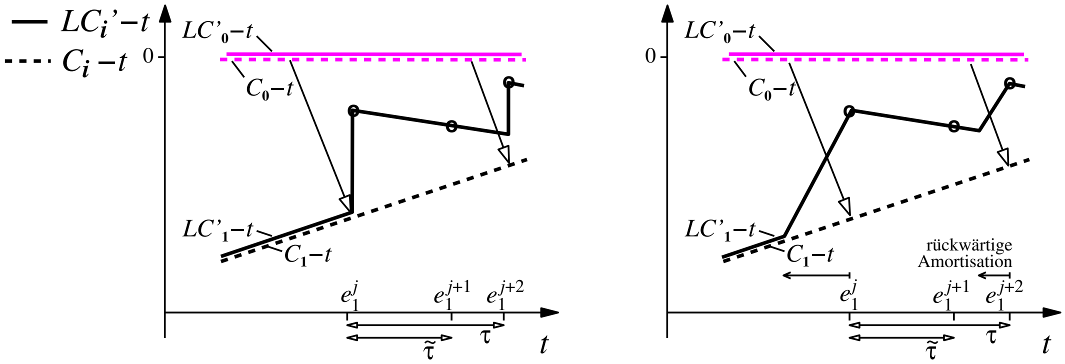


Abb. 5.2: Szenarium zur Fehlerabschätzung des Reglers $\mathcal{A}$, links *ohne* und rechts *mit* rückwärtiger Amortisation

Abb. 5.2 stellt den Verlauf von LC_1' dar. Hierbei gilt

$$\begin{aligned}
LC_1'(e_1^j) - t(e_1^j) &= LC_0'(e_0^j) - t(e_0^j) + \mu_{0,1} - m(e_1^j) \\
&= \mu_{0,1} - m(e_1^j) && (5.3) \\
\text{mit} \quad m(e_1^j) &= {\scriptstyle \text{Nachrichtenlaufzeit} \atop \scriptstyle \text{der empfangenen Nachricht}} = t(e_1^j) - t(e_0^j) \\
LC_1'(e_1^{j+2}) - t(e_1^{j+2}) &= \mu_{0,1} - m(e_1^{j+2}) && (5.4) \\
\text{mit} \quad m(e_1^{j+2}) &= {\scriptstyle \text{Nachrichtenlaufzeit} \atop \scriptstyle \text{der empfangenen Nachricht}} = t(e_1^{j+2}) - t(e_0^{j+2}) \\
\text{Steigung von } LC_1' - t \text{ nach } e_1^j &= \gamma_{\max}(1+\rho) \; - \; 1 && (5.5) \\
\text{Steigung von } LC_1' - t \text{ mit Amortisation vor } e_1^{j+2} &= \gamma_{\max}(1+\rho)(1+A) \; - \; 1 && (5.6)
\end{aligned}$$

Im folgenden soll der relative Fehler der geregelten logischen Uhr LC_1' im Vergleich zur exakten Zeit abgeschätzt werden, d.h. für

$$\epsilon^{LC_1'} = \left| \frac{LC_1'(e_1^l) - LC_1'(e_1^j)}{t(e_1^l) - t(e_1^j)} - 1 \right|$$

soll eine obere Schranke ermittelt werden.

Ohne rückwärtige Amortisation

ist der maximale Fehler durch zwei Sachverhalte bestimmt:

1. die Sprünge bei den Empfangsereignissen und
2. der durch $\gamma_{\max}$ und ρ bestimmte Verlauf zwischen zwei Sprüngen.

Im folgenden wird nur der Fehler bei zwei direkt aufeinanderfolgenden Ereignissen untersucht, da der maximale Fehler der Zeitbestimmung zwischen zwei beliebigen Ereignissen immer kleinergleich dem maximalen Fehler bei direkt aufeinanderfolgenden Ereignissen sein muß.

Der Fehler durch die Sprünge wirkt sich dann nur auf das Intervall aus, das vor dem Sprung liegt. Da sich innerhalb dieses Intervalls auch der Fehler durch $\gamma_{\max}$ und ρ entsteht, sind beide Fehlerarten zu addieren. Aus (5.3), (5.4) und (5.5) ergibt sich damit:

$$
\begin{aligned}
&\max_j \left| \frac{LC_1'(e_1^{j+1}) - LC_1'(e_1^j)}{t(e_1^{j+1}) - t(e_1^j)} - 1 \right| \\
\leq & \frac{\max_{\substack{j<l,\, e_1^j, e_1^l \text{ aufeinanderfolgende} \\ \text{Empfangsereignisse},\ \rho \in [-\rho_{\max}, \rho_{\max}]}} \left| \mu_{0,1} - m(e_1^l) - (\mu_{0,1} - m(e_1^j) + (\gamma_{\max}(1+\rho) - 1)[t(e_1^l) - t(e_1^j)]) \right|}{\min_j (t(e_1^{j+1}) - t(e_1^j))} \\
&+ \max_{\rho \in [-\rho_{\max}, \rho_{\max}]} |\gamma_{\max}(1+\rho) - 1| \\
\leq & \frac{\max_{j<l,\, e_1^j, e_1^l \substack{\text{aufeinanderfolgende} \\ \text{Empfangsereignisse}}} \left| m(e_1^j) - m(e_1^l) \right|}{\min_j (t(e_1^{j+1}) - t(e_1^j))} \\
&+ \frac{\max_{j<l,\, e_1^j, e_1^l \substack{\text{aufeinanderfolgende} \\ \text{Empfangsereignisse}}} (t(e_1^l) - t(e_1^j))}{\min_j (t(e_1^{j+1}) - t(e_1^j))} [(1-\gamma_{\max}) + \rho_{\max}\gamma_{\max}] \\
&+ [(1-\gamma_{\max}) + \rho_{\max}\gamma_{\max}]
\end{aligned} \tag{5.7}
$$

Diese Abschätzung gilt nicht für den ersten Sprung, da hierbei die Sprunghöhe durch die Uhrendifferenz und nicht durch die Gangabweichung seit dem letzten Sprung bestimmt wird. Der erste Summand in (5.7) ist durch den Quotient der Varianz der Nachrichtenlaufzeiten geteilt durch den minimalen Abstand zweier Ereignisse bestimmt. Dieser Wert liegt normalerweise im Bereich mehrerer Zehnerpotenzen und impliziert damit die Notwendigkeit, die rückwärtige Amortisation einzusetzen. Der zweite Summand in (5.7) besteht aus einem Produkt. Der erste Multiplikator, der den maximalen Abstand zweier aufeinanderfolgender Empfangsereignisse in Relation zum minimalen Abstand zweier beliebiger Ereignisse setzt, kann ebenfalls mehrere Zehnerpotenzen betragen. Er wird aber durch den zweiten Multiplikator relativiert, der durch $(1-\gamma_{\max})$ und $\rho_{\max}$ bestimmt ist und damit typischerweise 10^{-4} nicht übersteigt. Das Produkt kann natürlich immer noch wesentlich über der maximale tolerierbaren Fehlerrate ϵ (z.B. 10 %) liegen. Auch dieser Summand wird durch die rückwärtige Amortisation auf ein akzeptables Maß begrenzt werden. Der dritte Summand ist im Vergleich zu den ersten beiden vernachlässigbar.

Mit rückwärtiger Amortisation

ist der Fehler durch das Maximum der Beträge der Steigungen der beiden Kurvenstücke gegeben, also

$$
\begin{aligned}
\max_j \left| \frac{LC_1^A(e_1^{j+1}) - LC_1^A(e_1^j)}{t(e_1^{j+1}) - t(e_1^j)} - 1 \right| \leq & \max\{ \max_{\rho \in [-\rho_{\max}, \rho_{\max}]} (|\gamma_{\max}(1+\rho) - 1|)\ , \\
& \max_{\rho \in [-\rho_{\max}, \rho_{\max}]} (|\gamma_{\max}(1+\rho)(1+A) - 1|) \} && (5.8) \\
\leq & A(1+\rho_{\max}) + (1-\gamma_{\max}) + \rho_{\max} && (5.9)
\end{aligned}
$$

$\langle$[1] Herleitung:
Da $\gamma_{\max} < 1$ ist gilt für den ersten Term, daß das Maximum für $\rho = -\rho_{\max}$ eingenommen wird, d.h.:

$$
\begin{aligned}
\max_{\rho \in [-\rho_{\max}, \rho_{\max}]} (|\gamma_{\max}(1+\rho) - 1|) =& - (\gamma_{\max}(1 + (-\rho_{\max})) - 1) \\
=& (1-\gamma_{\max}) + \gamma_{\max}\rho_{\max} \\
\leq & (1-\gamma_{\max}) + \rho_{\max}
\end{aligned}
$$

[1] Nebenrechnungen sind in Winkelklammern $\langle\rangle$ gesetzt. Sie sind zwar für das Verständnis des Beweises irrelevant, sind aber hilfreich bei der Verifikation des Beweises.

Für den zweiten Term muß man unterscheiden:
1) $\gamma_{\max}(1+A) \leq 1$, dann Maximum bei $\rho = -\rho_{\max}$:

$$\begin{aligned}\max_{\rho\in[-\rho_{\max},\rho_{\max}]}(|\gamma_{\max}(1+\rho)(1+A) - 1|) &= -\left(\gamma_{\max}(1+(-\rho_{\max}))(1+A) - 1\right)\\ &= 1-\gamma_{\max}+\gamma_{\max}\rho_{\max} - A\gamma_{\max}(1-\rho_{\max})\\ &\leq (1-\gamma_{\max})+\rho_{\max}\end{aligned}$$

2) $\gamma_{\max}(1+A) \geq 1$, dann Maximum bei $\rho = +\rho_{\max}$:

$$\begin{aligned}\max_{\rho\in[-\rho_{\max},\rho_{\max}]}(|\gamma_{\max}(1+\rho)(1+A) - 1|) &= +\left(\gamma_{\max}(1+(+\rho_{\max}))(1+A) - 1\right)\\ &= -(1-\gamma_{\max})+\gamma_{\max}\rho_{\max} + A\gamma_{\max}(1+\rho_{\max})\\ &\leq \rho_{\max}+A(1+\rho_{\max})\end{aligned}$$

Und damit ist das Maximum der beiden Terme kleiner oder gleich der Summe aller (positiven) Summenden in den Einzelabschätzungen. q.e.d. $\rangle$[2]

Diese Abschätzung gilt – im Unterschied zur Abschätzung *ohne* Amortisation – auch für den Bereich vor dem ersten Vorstellen der LC_1^A. Da bei der Herleitung dieser Fehlerabschätzung eine Gangabweichung von C_0 keinen Einfluß gehabt hätte, und da dann das Szenarium der allgemeinste Fall ist, gilt diese Abschätzung generell, wenn ausschließlich der Regler $\mathcal{A}$ und die rückwärtige Amortisation zum Einsatz kommen.

Da normalerweise $\max(A)$ wesentlich größer ist als $1-\gamma_{\max}$ und $\rho_{\max}$, bestimmt der Amortisationsgradient A hauptsächlich den Fehler. Solange bei der Amortisation keine konvexe Hülle gebildet werden muß, ist $A = A_{\text{def}}$. Sobald aber aufgrund eines Sendeereignisse vor dem rückwärts amortisierten Empfangsereignis die konvexe Hülle gebildet werden muß, wird zwischen diesen beiden Ereignissen der Amortisationgradient A steiler. Es gilt dann

$$A \leq \frac{\text{maximale Sprunghöhe ohne Amortisation}}{\substack{\text{minimaler Abstand zwischen einem Sendeereignis}\\ \text{und einem nachfolgenden Empfangsereignis im gleichen Prozeß}}}. \tag{5.10}$$

Sendeereignisse, die die Bildung der konvexen Hülle auslösen, müssen immer zu einer Nachricht an einen dritten Prozeß gehören. Sobald durch erste Nachrichten die Uhren an die am meisten vorgehende angeglichen sind, gilt dann wieder eine durch die maximale Sprunghöhe bestimmte Abschätzung analog zu den ersten beiden Summanden in (5.7), aber die Relation wird nicht zum kürzesten Abstand zweier beliebiger Ereignisse gemacht, sondern zum kürzesten Abstand eines Empfangsereignisse und dem vorhergehenden Sendeereignis, also

$$\begin{aligned}A \leq \max\{A_{\text{def}}, &\frac{\max_{j<l,\, e_1^j, e_1^l \ \substack{\text{aufeinanderfolgende}\\ \text{Empfangsereignisse}}} \left|m(e_1^j) - m(e_1^l)\right|}{\min_{j<l,\, e_1^j \text{ Sendeereignis},\, e_1^l \text{ Empfangsereignis}}(t(e_1^l) - t(e_1^j))}\\ &+ \frac{\max_{j<l,\, e_1^j, e_1^l \ \substack{\text{aufeinanderfolgende}\\ \text{Empfangsereignisse}}} (t(e_1^l) - t(e_1^j))}{\min_{j<l,\, e_1^j \text{ Sendeereignis},\, e_1^l \text{ Empfangsereignis}}(t(e_1^l) - t(e_1^j))}[(1-\gamma_{\max})+\rho_{\max}\gamma_{\max}]\}\end{aligned}$$

Da der erste Summand trotz der Änderung im Divisor über der maximal tolerierbaren Fehlerrate ϵ liegen kann, muß die Abschätzung verschärft werden, indem man nur Empfangsereignisse betrachtet, deren Nachrichtenlaufzeiten höchstens $1+\epsilon/2$ der minimalen Nachrichtenlaufzeit $\tilde{\mu}$ beträgt[3]. Damit ergibt sich

$$A \leq \bar{A} \tag{5.11}$$

[2] siehe Fußnote 1 auf Seite 48

[3] μ stellt *eine* untere Schranke dar, während $\tilde{\mu}$ *das* Minimum ist, d.h. es gilt $\mu \leq \tilde{\mu}$.

mit

$$\bar{A} = \max\{A_{\text{def}}, \frac{\epsilon}{2}\frac{\tilde{\mu}}{\min_{\substack{j<l,\, e_1^j \text{ Sendeereignis},\\ e_1^l \text{ Empfangsereignis},\, m(e_1^l)\le(1+\epsilon/2)\tilde{\mu}}}(t(e_1^l)-t(e_1^j))}$$

$$+\frac{\max_{\substack{j<l,\, e_1^j, e_1^l \text{ aufeinanderfolgende}\\ \text{Empfangsereignisse mit } m(e)\le(1+\epsilon/2)\tilde{\mu}}}(t(e_1^l)-t(e_1^j))}{\min_{\substack{j<l,\, e_1^j \text{ Sendeereignis},\\ e_1^l \text{ Empfangsereignis},\, m(e_1^l)\le(1+\epsilon/2)\tilde{\mu}}}(t(e_1^l)-t(e_1^j))}[(1-\gamma_{\max})+\rho_{\max}\gamma_{\max}]\} \tag{5.12}$$

$$= \max\{A_{\text{def}}, \epsilon\tilde{\kappa}_a\tfrac{1}{2}(1+\tilde{\kappa}_b)\}$$

und

$$\tilde{\kappa}_a = \frac{\tilde{\mu}}{\min_{\substack{j<l,\, e_1^j \text{ Sendeereignis},\\ e_1^l \text{ Empfangsereignis},\, m(e_1^l)\le(1+\epsilon/2)\tilde{\mu}}}(t(e_1^l)-t(e_1^j))} \tag{5.13}$$

$$\tilde{\kappa}_b = \frac{\max_{\substack{j<l,\, e_1^j, e_1^l \text{ aufeinanderfolgende}\\ \text{Empfangsereignisse mit } m(e)\le(1+\epsilon/2)\tilde{\mu}}}(t(e_1^l)-t(e_1^j))}{\tilde{\mu}}\frac{(1-\gamma_{\max})+\rho_{\max}\gamma_{\max}}{2\epsilon} \tag{5.14}$$

Der Fehler ist kleiner als ϵ, wenn $\tilde{\kappa}_a$ kleiner als 1 ist, d.h. wenn die Abstände zwischen Empfangsereignissen und vorhergehenden Sendeereignissen größer als die minimale Nachrichtenlaufzeiten sind, und wenn $\tilde{\kappa}_b$ kleiner als 1/2 ist, d.h. wenn das Verhältnis der Abstände zwischen zwei aufeinanderfolgenden Empfangsereignissen (mit Nachrichtenlaufzeiten kleiner als $(1+\epsilon)\tilde{\mu}$) zu der minimalen Nachrichtenlänge $\tilde{\mu}$ kleiner ist als das Verhältnis von 2ϵ zur Summe von $1-\gamma_{\max}$ und $\rho_{\max}$ ($\gamma_{\max}$ kann hierbei vernachlässigt werden). Die folgenden Beispiele geben zwei sinnvolle Szenarien, die zeigen, daß in der Praxis zu erwarten ist, daß mit einer ausreichenden Beschränkung des Fehlers zu rechnen ist.

Beispiel 1: Für $1-\gamma_{\max} = 2\cdot10^{-5}$, $\rho = 10^{-5}$, $A_{\text{def}} = 0.005$, $\epsilon = 5\,\%$, $\tilde{\kappa}_a = \frac{5\mu s}{3\mu s} = 1.67$, und $\tilde{\kappa}_b = \frac{10000\mu s}{5\mu s}\frac{3\cdot10^{-5}}{2\cdot0.05} = 0.6$ ergibt $A \le \max\{0.005, 0.067\} = 6.7\,\%$.

Beispiel 2: Für $1-\gamma_{\max} = 2\cdot10^{-4}$, $\rho = 10^{-4}$, $A_{\text{def}} = 0.005$, $\epsilon = 5\,\%$, $\tilde{\kappa}_a = \frac{500\mu s}{300\mu s} = 1.67$, und $\tilde{\kappa}_b = \frac{0.1s}{500\mu s}\frac{3\cdot10^{-4}}{2\cdot0.05} = 0.6$ ergibt $A \le \max\{0.005, 0.067\} = 6.7\,\%$.

Am Anfang sind die Sprünge begrenzt durch die maximale Uhrendifferenz. Dies ergibt:

$$A \le \check{A} \tag{5.15}$$

mit

$$\check{A} = \max\{A_{\text{def}}, \check{\kappa}\} \tag{5.16}$$

und

$$\check{\kappa} = \frac{\max_{i,k,t}(C_i(t)-C_k(t))}{\min_{j<l,\, e_1^j \text{ Sendeereignis},\, e_1^l \text{ Empfangsereignis}}(t(e_1^l)-t(e_1^j))} \tag{5.17}$$

Bei Systemen, bei denen $\check{\kappa}$ größer als die gewünschte maximale Ungenauigkeit (z.B. 5%) ist, sollte die geregelte logische Uhr zusammen mit einer einmaligen Uhrensynchronisation am Beginn der zu beobachtenden Anwendung eingesetzt werden. Damit ist der maximale Fehler dann durch (5.9) und (5.12) limitiert.

5.3 Charakteristik der Regler $\mathcal{B}$ und $\mathcal{D}$

Der Regler $\mathcal{B}$ soll anhand des folgenden Szenariums analysiert werden. Zuerst wird der Verlauf der geregelten logischen Uhr LC' bestimmt, um anschließend eine Fehlerabschätzung durchführen zu können.

Szenarium 2 *Die Ereignisse e_0^j und e_1^j seien jeweils das Senden und Empfangen zweier gegenläufiger Nachrichten zwischen den Prozessen 0 und 1. Die Nachrichtenlaufzeiten seien Null. Der zeitliche Abstand der Nachrichten sei τ. Die Uhr von Prozeß 0 geht exakt, d.h. $C_0(j) = j\tau$. Die Uhr von Prozeß 1 geht zum Zeitpunkt Null um D nach und sie geht um ρ schneller als C_0, d.h. $C_1(j) = j\tau(1+\rho) - D$. Das Szenarium ist in Abb. 5.3 dargestellt und entspricht dem linken Diagramm in Abb. 4.5.*

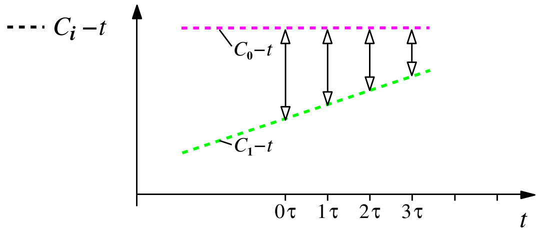


Abb. 5.3: Szenarium zur Fehlerabschätzung

Der gegenseitige Nachrichtaustausch ergibt, daß die geregelten logischen Uhren der beiden Prozesse and den Zeitpunkten $j\tau$ identisch sind, d.h. $LC_0'(e_0^j) = LC_1'(e_1^j)$. $LC'(j)$ sei daher eine Abkürzung für $LC_0'(e_0^j)$ und $LC_1'(e_1^j)$. Für LC' gilt:

$$\begin{aligned}
LC'(0) &= 0 \\
LC'(j) &= \max(LC'(j-1) + \gamma_{j-1}(C_1(j) - C_1(j-1)), C_0(j)) \\
&= \max(LC'(j-1) + \gamma_{j-1}\tau(1+\rho), j\tau) \qquad (5.18)
\end{aligned}$$

mit

$$\begin{aligned}
\gamma_0 &= \gamma_{\max} \\
\gamma_{j-1} &= \gamma_{\max}\left[1 - \left(\frac{LC'(j-1) - C_0(j-1)}{D}\right)^2\right] \\
&= \gamma_{\max}\left[1 - \left(\frac{LC'(j-1) - (j-1)\tau}{D}\right)^2\right] \quad \text{für } \gamma_{\max}(1+\rho) > 1 \qquad (5.19)
\end{aligned}$$

Die ersten beiden Formeln definieren die Strecke des Regelkreises, die letzten beiden Formeln stellen den Regler dar. Damit der Regler $\mathcal{B}$ zum Einsatz kommt, muß $\gamma_{\max}(1+\rho) > 1$ sein, d.h.

$$\rho \quad > \quad \frac{1 - \gamma_{\max}}{\gamma_{\max}} \qquad (5.20)$$

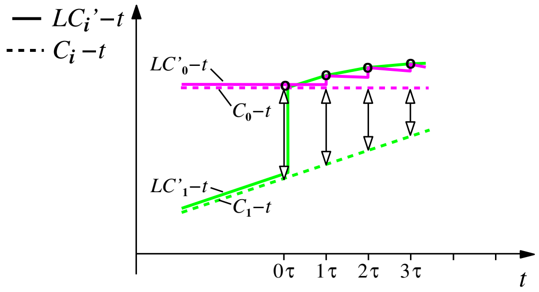


Abb. 5.4: Die geregelte logische Uhr in obigem Szenarium

Abb. 5.4 stellt den Verlauf der geregelten logischen Uhr schematisch dar. Im Gegensatz zu Abb. 4.8 ist hier nun die Regelung $\mathcal{B}$ und nicht $\mathcal{A}$ für die Begrenzung des Überschwingens verantwortlich. Um den Verlauf von LC' in Abhängigkeit der Größen τ, ρ, $\gamma_{\max}$ und D zu analysieren, wird zuerst eine transformierte Kurve analysiert, und anschließend die Erkenntnisse wieder auf LC' übertragen. Die Transformation ist definiert durch:

$$O_j = LC'(j) - C_0(j) \qquad \text{Offset} \tag{5.21}$$

$$\Omega_j = \frac{O_j}{D} \qquad \text{Relation Offset zu } D \tag{5.22}$$

$$\theta = \frac{\tau}{D} \qquad \text{Relation } \tau \text{ zu } D \tag{5.23}$$

$$R = \gamma_{\max}(1+\rho) \qquad \text{relativierte Uhrenabweichung} \tag{5.24}$$

Die Bedingung (5.20) für den Einsatz des Reglers $\mathcal{B}$ ist äquivalent zu $R > 1$. Das Bildungsgesetz für LC' ergibt dann folgende transformierte Darstellung:

$$\begin{aligned} \Omega_0 &= 0 \\ \Omega_j &= \max(0, \Omega_{j-1} - \Omega_{j-1}^2\theta R + \theta(R-1)) \end{aligned}$$

⟨[4] Herleitung:

$$\begin{aligned}
LC'(j) &= \max(LC'(j-1) + \gamma_{\max}\left[1 - \left(\frac{LC'(j-1)-(j-1)\tau}{D}\right)^2\right]\tau(1+\rho), j\tau) \\
O_j &= LC'(j) - C_0(j) = LC'(j) - j\tau \\
&= LC'(j) - (j-1)\tau - \tau \\
&= \max(LC'(j-1) - (j-1)\tau - \tau + \gamma_{\max}\left[1 - \left(\frac{LC'(j-1)-(j-1)\tau}{D}\right)^2\right]\tau(1+\rho), j\tau - j\tau) \\
&= \max(O_{j-1} - \tau + R\tau\left[1 - \left(\frac{O_{j-1}}{D}\right)^2\right], 0) \\
&= \max(O_{j-1} - \tau + R\tau - R\tau\left(\frac{O_{j-1}}{D}\right)^2, 0) \\
&= \max(O_{j-1} + (R-1)\tau - R\tau\left(\frac{O_{j-1}}{D}\right)^2, 0) \\
\Omega_j &= \max(\Omega_{j-1} + (R-1)\theta - R\theta\Omega_{j-1}^2, 0) \qquad \rangle^4
\end{aligned}$$

[4]siehe Fußnote 1 auf Seite 48

Eine Grenzwertbestimmung unter der Voraussetzung, daß Ω_j für $j \to \infty$ konvergiert (dies wird später analysiert), ergibt als Grenzwert

$$\Omega_\infty = \sqrt{\frac{R-1}{R}} \tag{5.25}$$

Logisch gesehen sei Ω_∞ definiert als $\sqrt{\frac{R-1}{R}}$. Die folgende Transformation

$$\begin{aligned} o_j &= \frac{\Omega_j}{\Omega_\infty} \qquad (5.26)\\ \text{bzw. } o_j &= \frac{LC'(j) - C_0(j)}{D\Omega_\infty} \end{aligned}$$

ergibt

$$\begin{aligned} o_0 &= 0 \\ o_j &= \max(o_{j-1} + f - f o_{j-1}^2, 0) \qquad (5.27)\\ \text{mit } f &= \theta\sqrt{R(R-1)} \\ &= \frac{\tau}{D}\sqrt{\gamma_{\max}(1+\rho)(\gamma_{\max}(1+\rho)-1)} \\ &= \frac{\tau}{D}\sqrt{[\rho - (1-\gamma_{\max})(1+\rho)]\,\gamma_{\max}(1+\rho)} \qquad (5.28)\\ &\simeq \frac{\tau}{D}\sqrt{\rho - (1-\gamma_{\max})} \text{ für } \rho \simeq 0 \text{ und } \gamma_{\max} \simeq 1 \qquad (5.29) \end{aligned}$$

⟨[5] Herleitung für Ω_∞:

$$\begin{aligned} \Omega_{j-1} &= \Omega_j \\ \Omega_{j-1} &= \Omega_{j-1} - \Omega_{j-1}^2\theta R + \theta(R-1) \\ \Omega_{j-1}^2\theta R &= \theta(R-1) \\ \Omega_{j-1}^2 &= \frac{R-1}{R} \end{aligned}$$

Herleitung für o_j:

$$\begin{aligned} \frac{\Omega_j}{\Omega_\infty} &= \max(\frac{\Omega_{j-1}}{\Omega_\infty} + \frac{(R-1)\theta}{\Omega_\infty} - R\theta\Omega_\infty\frac{\Omega_{j-1}^2}{\Omega_\infty^2}, 0) \\ o_j &= \max(o_{j-1} + \theta\sqrt{R(R-1)} - \theta\sqrt{R(R-1)}o_{j-1}^2, 0) \end{aligned}$$ ⟩[5]

Die letzte Transformation ist die Bildung einer inversen Funktion, von der später gezeigt wird, daß sie in einigen Fällen gegen Null konvergiert:

$$v_j = 1 - o_j \tag{5.30}$$

Dies ergibt

$$\begin{aligned} v_0 &= 1 \\ v_j &= \min(v_{j-1}\left[1 - f(2 - v_{j-1})\right], 1) \qquad (5.31) \end{aligned}$$

[5]siehe Fußnote 1 auf Seite 48

$\langle^6$ Herleitung:

$$\begin{aligned} v_j &= 1 - \max(o_{j-1} + f - f o_{j-1}^2, 0) \\ &= \min(1 - o_{j-1} - f + f o_{j-1}^2, 1) \\ &= \min(v_{j-1} - f + f(1 - v_{j-1})^2, 1) \\ &= \min(v_{j-1} - f + f - 2 f v_{j-1} + f v_{j-1}^2, 1) \qquad \rangle^6 \end{aligned}$$

Die Rücktransformation ergibt

$$LC'(j) = C_0(j) + o_j D \sqrt{\frac{\rho - (1 - \gamma_{\max})(1 + \rho)}{\gamma_{\max}(1 + \rho)}} \tag{5.32}$$

$$\overset{\text{für } \rho \simeq 0 \wedge \gamma_{\max} \simeq 1}{\simeq} C_0(j) + o_j D \sqrt{\rho - (1 - \gamma_{\max})} \tag{5.33}$$

$\langle^6$ Herleitung:

$$\begin{aligned} LC'(j) &= C_0(j) + O_j \\ &= C_0(j) + D\Omega_j \\ &= C_0(j) + D\Omega_\infty o_j = C_0(j) + D\sqrt{\frac{R-1}{R}} o_j \\ R - 1 &= \gamma_{\max}(1 + \rho) - 1 \\ &= (1 - (1 - \gamma_{\max}))(1 + \rho) - 1 \\ &= 1 + \rho - (1 - \gamma_{\max})(1 + \rho) - 1 = \rho - (1 - \gamma_{\max})(1 + \rho) \qquad \rangle^6 \end{aligned}$$

Die Transformation der vierparametrigen Folgenschar $(\{LC(j)\}_{j=0,1,\ldots}, \tau, \rho, \gamma_{\max}, D)$ in die einparametrige Schar $(\{o_j\}_{j=0,1,\ldots}, f)$ vereinfacht die Fallunterscheidungen bei der Analyse dieser Folgenschar. Der folgende Satz analysiert die Entwicklung der Folge $\{o_j\}_{j=0,1\ldots}$ für alle möglichen Werte von f:

[6] siehe Fußnote 1 auf Seite 48

Satz 10 *Es gilt*

$$
\begin{array}{lll}
f \in (0, \frac{1}{2}] & \Longrightarrow \lim_{j\to\infty} o_j = 1,\ \{o_j\}_{j=0,1,...} \text{ ist streng monoton steigend} & (5.34)\\
f \in (\frac{1}{2}, \frac{\sqrt{5}-1}{2}) & \Longrightarrow \lim_{j\to\infty} o_j = 1,\ \{|1-o_j|\}_{j=0,1,...} \text{ ist monoton fallend} & \\
 & \quad \text{und } o_0 = 0, o_1 = f, o_{j,\,j=1,2,...} \in [f, 1.0052] & (5.35)\\
f = \frac{\sqrt{5}-1}{2} & \Longrightarrow o_0 = 0,\ o_1 = f,\ o_{j,\,j=1,2,...} = 1 & (5.36)\\
f \in (\frac{\sqrt{5}-1}{2}, 1) & \Longrightarrow \lim_{j\to\infty} o_j = 1,\ \{|1-o_j|\}_{j=0,1,...} \text{ ist monoton fallend} & \\
 & \quad \text{und } o_j > 1 \text{ für } j \text{ gerade, } o_j < 1 \text{ für } j \text{ ungerade} & \\
 & \quad \text{und } o_0 = 0, o_1 = f, o_{j,\,j=1,2,...} \in [f, \sqrt{32/27}] & (5.37)\\
f = 1 & \Longrightarrow o_0 = 0,\ o_{j,\,j=1,2,...} = 1 & (5.38)\\
f \in (1, 1.224635] & \Longrightarrow \{o_j\}_{j=0,1,...} \text{ „konvergiert periodisch“ mit } L = 2 & \\
 & \quad o_{j,\,j\ gerade} \in [0,1),\ o_{j,\,j\ ungerade} \in (1, 1.41276] & (5.39)\\
f \in [1.224636, 1.272003] & \Longrightarrow \{o_j\}_{j=0,1,...} \text{ „konvergiert periodisch“ mit } L = 4 & \\
 & \quad o_{j,\,j\ gerade} \in [0, 0.61264], & \\
 & \quad o_{j,\,j\ ungerade} \in [1.22464, 1.46706] & (5.40)\\
f \in [1.272004, 1.272858] & \Longrightarrow \{o_j\}_{j=0,1,...} \text{ „konvergiert periodisch“ mit } L = 8 & \\
 & \quad o_{j,\,j\ gerade} \in [0, 0.48591],\ o_{j,\,j\ ungerade} \in [1.2720, 1.4672] & (5.41)\\
f \in [1.272859, 1.277486] & \Longrightarrow o_{j,\,j=0,8,16,...} = 0,\ o_{j,\,j\neq 0,8,16,...} \in [0, 1.46720] & (5.42)\\
f \in [1.277487, \sqrt{2}) & \Longrightarrow o_{j,\,j=0,4,8,...} = 0,\ o_{j,\,j\neq 0,4,8,...} \in [0, 1.51338] & (5.43)\\
f \in [\sqrt{2}, \infty) & \Longrightarrow o_{j,\,j=0,2,4,...} = 0,\ o_{j,\,j=1,3,5,...} = f & (5.44)
\end{array}
$$

„Periodische Konvergenz“ in (5.39) bis (5.41) bedeutet, daß die L Folgen $o_{j,\,j=l,l+L,l+2L,...}$ für jedes feste $l \in \{0,1,..,L\}$ konvergieren.

Der Verlauf von $\{o_j\}_{j=0,1,...}$ in den verschiedenen Fällen ist exemplarisch in den Abbildungen 5.5 bis 5.7 auf Seite 56 dargestellt.

Beweis: Der Beweis ist weitgehend in geschlossener Darstellung möglich. Nur die Fälle $f \in (1, \sqrt{2})$ wurden numerisch behandelt. Durch die Reduktion auf eine einparametrige Folgenschar ist dies ausreichend. Die Einzelaussagen des Satzes können folgendermaßen bewiesen werden:

(5.34)-(5.37) Es sei $0 < \varepsilon \leq f \leq 1 - \varepsilon < 1$.

Lemma: Es gilt $|v_j| \leq (1-\varepsilon)|v_{j-1}|$ und $|v_{j-1}| \leq 1 - f$ für $j \geq 2$

Beweis: Für $j = 2$ gilt die rechte Ungleichung der Behauptung, da $v_1 = 1 - f$. Um die linke Ungleichung zu beweisen, muß man die rechte voraussetzen und nachfolgende Fälle unterscheiden. Vollständige Induktion ergibt dann die Gültigkeit beider Ungleichungen für jedes $j \geq 2$.

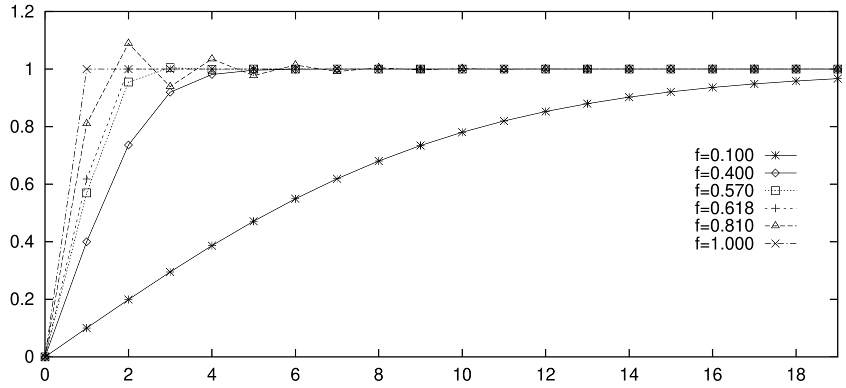


Abb. 5.5: o_j über j, konvergierende Fälle

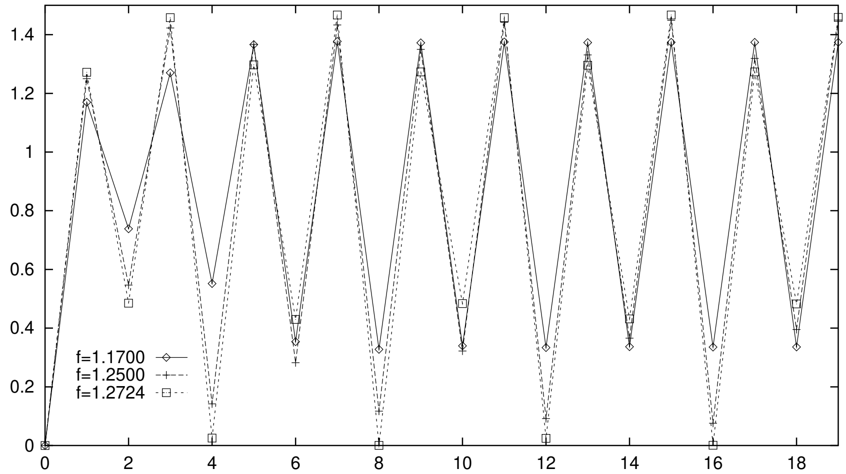


Abb. 5.6: o_j über j, periodisch konvergierend

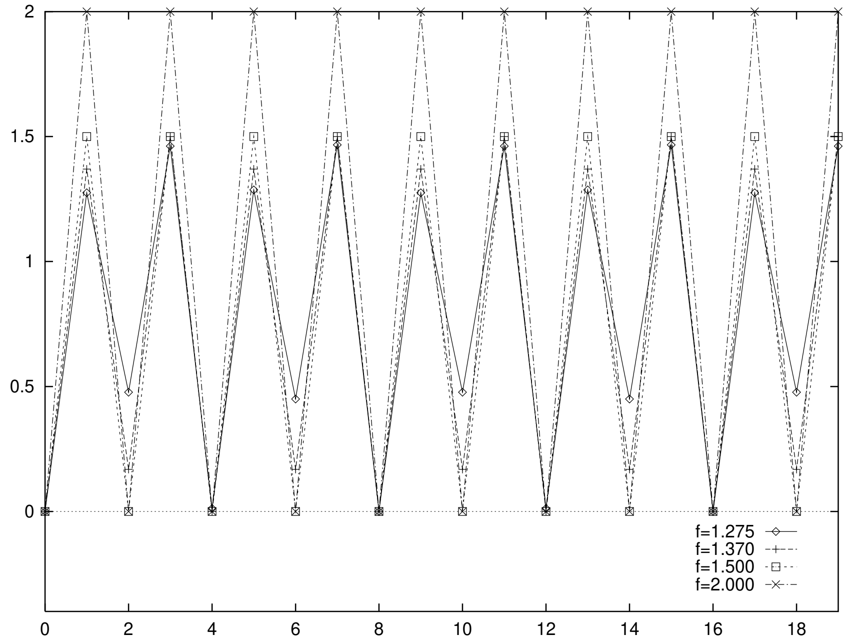


Abb. 5.7: o_j über j, periodische Fälle

•Es sei $v_{j-1} \geq 0$. Zu zeigen ist $-(1-\varepsilon)v_{j-1} \leq v_j \leq (1-\varepsilon)v_{j-1}$.
Die rechte Seite ergibt sich folgendermaßen:

$$\begin{aligned}
v_j &= \underbrace{v_{j-1}}_{\geq 0}(1-2f+f\underbrace{v_{j-1}}_{\leq 1-f}) \\
&\leq v_{j-1}(1-2f+f(1-f)) \\
&= v_{j-1}(1-\underset{\geq\varepsilon}{f}-\underset{\geq 0}{f^2}) \\
&\leq v_{j-1}(1-\varepsilon)
\end{aligned}$$

und die linke Seite ergibt sich aus:

$$\begin{aligned}
v_j &= \underbrace{v_{j-1}}_{\geq 0}(1-2f+f\underbrace{v_{j-1}}_{\geq 0}) \\
&\geq v_{j-1}(1-2f) = v_{j-1}(1-\underset{\leq 1-\varepsilon}{f}-\underset{\leq 1}{f}) \\
&\geq v_{j-1}(-1+\varepsilon)
\end{aligned}$$

•Es sei $v_{j-1} \leq 0$. Zu zeigen ist $(1-\varepsilon)v_{j-1} \leq v_j \leq -(1-\varepsilon)v_{j-1}$.
Die rechte Seite ergibt sich aus:

$$\begin{aligned}
v_j &= v_{j-1}(1-2f+fv_{j-1}) \\
&= \underbrace{(-v_{j-1})}_{\geq 0}(-1+2f+f\underbrace{(-v_{j-1})}_{\leq 1-f}) \\
&\leq (-v_{j-1})(-1+2f+f(1-f)) \\
&= (-v_{j-1})(1-\underbrace{(1-f)}_{\geq\varepsilon}-\underbrace{(1-f)^2}_{\geq 0}) \\
&\leq (-v_{j-1})(1-\varepsilon)
\end{aligned}$$

Die linke Seite ergibt sich aus:

$$\begin{aligned}
v_j &= \underbrace{(-v_{j-1})}_{\geq 0}(-1+2f+f\underbrace{(-v_{j-1})}_{\geq 0}) \\
&\geq (-v_{j-1})(-1+2f) = (-v_{j-1})(-1+\underset{\geq\varepsilon}{f}+\underset{\geq 0}{f}) \\
&\geq (-v_{j-1})(-1+\varepsilon)
\end{aligned}$$

□

Damit ist $\{|v_j|\}_{j=0,1,\ldots}$ mit einer geometrischen Folge mit dem Faktor $1-\varepsilon$ begrenzt und damit gegen Null konvergierend und damit $\lim_{j\to\infty} o_j = 1$.

(5.34) $\{o_j\}_{j=0,1,\ldots}$ ist streng monoton steigend, wenn mittels vollständiger Induktion gezeigt ist, daß $v_j > 0$ für alle $j = 0, 1 \ldots$.

$$\begin{aligned}
v_0 &= 1 > 0 \\
v_j &= \min(\underset{>0}{v_{j-1}}\underbrace{[1-\underbrace{\underset{<\frac{1}{2}}{f}\underbrace{(2-\underset{>0}{v_{j-1}})}_{<2}}_{<1}]}_{>0}, 1) > 0
\end{aligned}$$

(5.35) Die Konvergenzaussage ist mit obigem Lemma vollständig bewiesen. Die Grenzen wurden numerisch ermittelt. Es wurde numerisch bestätigt, daß für $0.5 \leq f \lesssim 0.524$ erstmals v_4 negativ wird und $v_4 \geq -0.0002$ ist und, daß für $0.524 \lessapprox f(\sqrt{5}-1)/2$ erstmals v_3 negativ wird und $v_3 \geq -0.0053$ ist. Die schon gezeigte Monotonie ergibt, daß damit alle $v_{j,j=0,1,\ldots} \geq -0.0053$ für $f \in (0.5, (\sqrt{5}-1)/2$.

(5.36) ergibt sich durch direkte Berechnung der Folge $\{o_j\}_{j=0,1,\ldots}$.

(5.37) Das alternierende Verhalten von $\{o_j\}_{j=0,1,\ldots}$ wird gezeigt, indem die äquivalente Aussage für $\{v_j\}_{j=0,1,\ldots}$ gezeigt wird, d.h. daß $v_{j,j=1,3,5\ldots} > 0$ und $v_{j,j=2,4,6\ldots} < 0$. Dies ergibt sich mit vollständiger Induktion ausgehend von $v_1 = 1-f > 0$.

$$\begin{aligned}
v_{j-1} &> 0 \wedge j \text{ gerade} \\
\Longrightarrow v_j &= v_{j-1}(1-2f+fv_{j-1}) \\
&\overset{v_{j-1}\leq 1-f}{\leq} v_{j-1}(1-f-f^2) \\
&= \underbrace{-v_{j-1}}_{>0}\underbrace{(f-\frac{\sqrt{5}-1}{2})}_{>0}\underbrace{(f+\frac{\sqrt{5}+1}{2})}_{>0} < 0
\end{aligned}$$

$$\begin{aligned}
v_{j-1} &< 0 \wedge j \text{ ungerade} \\
\Longrightarrow v_j &= v_{j-1}(1-2f+fv_{j-1}) \\
&= \underbrace{-v_{j-1}}_{>0}(\underbrace{2f-1}_{>0}+f\underbrace{(-v_{j-1})}_{>0}) > 0
\end{aligned}$$

Nun sind nur noch die Schranken für o_j für $j \geq 2$ zu zeigen. Die untere Schranke $o_j \geq f$ ergibt sich aus der schon gezeigten Monotonie von $\{|1-o_j|\}_{j=0,1\ldots}$ und daß $o_1 = f$. Die obere Schranke für $\{o_j\}_{j\geq 2}$ ergibt sich aus dem negativsten Wert von $\{1-o_j\}_{j\geq 2}$. Aufgrund der gezeigten Monotonie genügt es zu zeigen, daß für den ersten negativen Wert von $1-o-j$ ($= v_j$) die Beschränkung $v_j \geq 1-\sqrt{32/27}$ ($\simeq -0.08867$) gilt. Im folgenden wird gezeigt, daß v_2 negativ ist, und daß $v_2 \geq 1-\sqrt{32/27}$.

$$\begin{aligned}
v_1 &= 1-f \\
v_2 &= v_1(1-2f+fv_1) \\
&= 1-2f+f^3
\end{aligned}$$

Da der Term $(1-2f+f^3)$ im Intervall $f \in [\frac{\sqrt{5}-1}{2}, 1]$ bei $f = \sqrt{2/3}$ sein einziges Minimum besitzt und an den Randpunkten des Intervalls Null ist, gilt:

$$v_2|_{f=\sqrt{2/3}} = 1-\sqrt{32/27} \leq v_2 \leq 0$$

(5.38) ergibt sich durch direkte Berechnung der Folge $\{o_j\}_{j=0,1,\ldots}$.

(5.39) - (5.43) Diese Fälle wurden mittels numerischer Berechnung der mit f parametrisierten Folgenschar $\{o_j\}_{j=0,1,\ldots}$ für hinreichend viele verschiedene Werte von f ermittelt.

(5.44)

$$\begin{aligned}
v_2 &= \min(v_1[1-f(2-v_1)], 1) \\
&= \min(\underbrace{\underbrace{(f-1)}_{\geq\sqrt{2}-1}\underbrace{[-1+f+f^2]}_{\geq\sqrt{2}+1}}_{\geq 1}, 1) = 1
\end{aligned}$$

Damit ist $v_2 = v_0$ und damit ist die Folge periodisch mit der Periodenlänge 2.

Der Fall (5.42) ist periodisch, da o_8 wieder den Anfangswert $o_0 = 0$ erreicht, da in diesem Fall in (5.27) der erste Term der Maximumbildung bei o_8 negativ wird, und daher o_8 auf Null gesetzt wird. Dies ergab sich daraus, daß die geregelte logische Uhr LC' gegenüber der Prozessoruhr C nicht nachgehen darf (s. letzter Term in (4.10) und (4.11)). In den Fällen (5.43) und (5.44) gilt das gleiche für o_4 bzw. o_2.

□

Die für die nachfolgende Fehleranalyse wesentlichen Aussagen des Satzes 10 lassen sich folgendermaßen zusammenfassen:

Satz 11 *Es gilt*

$$f \in (0, \tfrac{1}{2}] \implies \lim_{j\to\infty} o_j = 1,\ \{o_j\}_{j=0,1,\ldots} \text{ ist streng monoton steigend} \quad (5.45)$$

$$f \in (\tfrac{1}{2}, 1] \implies \lim_{j\to\infty} o_j = 1,,\ \{|1 - o_j|\}_{j=0,1,\ldots} \text{ ist monoton fallend}$$

$$o_j \in [0, 1.08867] \quad (5.46)$$

$$f \in (1, 1.224635] \implies o_j \in [0, 1.41276] \quad (5.47)$$

$$f \in (1.224635, \sqrt{2}) \implies o_j \in [0, 1.51338] \quad (5.48)$$

$$f \in [\sqrt{2}, \infty) \implies o_{j,\,j=0,2,4,\ldots} = 0,\ o_{j,\,j=1,3,5,\ldots} = f \quad (5.49)$$

Unter Anwendung der Rücktransformation (5.33) und (5.29) ergibt dies:

Satz 12 *Ist* $\gamma_{\max}(1+\rho) > 1$ *dann gelten für*

$$f = \frac{\tau}{D}\sqrt{[\rho - (1-\gamma_{\max})(1+\rho)]\,\gamma_{\max}(1+\rho)} \simeq \frac{\tau}{D}\sqrt{\rho - (1-\gamma_{\max})} \quad (5.28, 5.29)$$

$$O_\infty = D\sqrt{\frac{\rho - (1-\gamma_{\max})(1+\rho)}{\gamma_{\max}(1+\rho)}} \simeq D\sqrt{\rho - (1-\gamma_{\max})} \quad (5.32, 5.33)$$

$$fO_\infty = \tau\,[\rho - (1-\gamma_{\max})(1+\rho)] \simeq \tau\,[\rho - (1-\gamma_{\max})] \quad (5.50)$$

folgender Verlauf von $LC' - C_0$

$$f \in (0, \tfrac{1}{2}] \implies \lim_{j\to\infty} (LC'(j) - C_0(j)) = O_\infty \text{ und}$$

$$LC'(j) - C_0(j) > LC'(j-1) - C_0(j-1) \quad (5.51)$$

$$f \in (\tfrac{1}{2}, 1] \implies \lim_{j\to\infty} (LC'(j) - C_0(j)) = O_\infty, \quad (5.52)$$

$$,\ \{|LC'(j) - C_0(j) - O_\infty|\}_{j=0,1,\ldots} \text{ ist monoton fallend,} \quad (5.53)$$

$$0 \le LC'(j) - C_0(j) \le 1.08867\, O_\infty \quad (5.54)$$

$$f \in (1, 1.224635] \implies 0 \le LC'(j) - C_0(j) \le 1.41276\, O_\infty \quad (5.55)$$

$$f \in (1.224635, \sqrt{2}) \implies 0 \le LC'(j) - C_0(j) \le 1.51338\, O_\infty \quad (5.56)$$

$$f \in [\sqrt{2}, \infty) \implies LC'(j) - C_0(j) = 0 \text{ für } j \text{ gerade}$$

$$LC'(j) - C_0(j) = fO_\infty \text{ für } j \text{ ungerade} \quad (5.57)$$

Die Approximationen gelten, wenn $\rho \simeq 0$ *und* $\gamma_{\max} \simeq 1$ *ist.*

Die exakte Form dieses Satzes ergibt sich mit (5.28) und (5.32) aus (5.45) bis (5.49). Die Approximationen weichen von der exakten Form des Satzes für $\rho \leq 10^{-2}$, $1-\gamma_{\max} \leq 10^{-2}$ und $1-\gamma_{\max} \leq \rho/2$ ebenfalls nur um höchstens 1 % ab.

Satz 13 *Im Szenarium 2 auf S. 51 gelten für $\gamma_{\max}(1+\rho) > 1$ folgende Fehlerabschätzungen für die lokale Bestimmung des Zeitintervalls zwischen zwei um $n\tau$ auseinanderliegende Ereignisse:*

$$f \in (0, \tfrac{1}{2}] \implies \left|\tfrac{LC'(j+n)-LC'(j)}{n\tau} - 1\right| \leq \rho - (1-\gamma_{\max}) \tag{5.58}$$

$$f \in (\tfrac{1}{2}, 1] \implies \left|\tfrac{LC'(j+n)-LC'(j)}{n\tau} - 1\right| \leq \min(1, \frac{2.17734}{n})(\rho - (1-\gamma_{\max})) \tag{5.59}$$

$$f \in (1, 1.224635] \implies \left|\tfrac{LC'(j+n)-LC'(j)}{n\tau} - 1\right| \leq \frac{1.41276}{n}(\rho - (1-\gamma_{\max})) \tag{5.60}$$

$$f \in (1.224635, \sqrt{2}) \implies \left|\tfrac{LC'(j+n)-LC'(j)}{n\tau} - 1\right| \leq \frac{1.2358}{n}(\rho - (1-\gamma_{\max})) \tag{5.61}$$

$$f \in [\sqrt{2}, \infty) \implies \left|\tfrac{LC'(j+n)-LC'(j)}{n\tau} - 1\right| \leq \rho - (1-\gamma_{\max}) \tag{5.62}$$

für

$$f = \frac{\tau}{D}\sqrt{[\rho - (1-\gamma_{\max})(1+\rho)]\,\gamma_{\max}(1+\rho)} \simeq \frac{\tau}{D}\sqrt{\rho - (1-\gamma_{\max})} \tag{5.28, 5.29}$$

Beweis:

(5.58) Zuerst wird die Gültigkeit der Fehlerabschätzung für $n = 1$, d.h. für zwei aufeinanderfolgende Ereignisse, gezeigt:

$$\begin{aligned}
\left|\frac{LC'(j) - LC'(j-1)}{\tau} - 1\right| &= \left|\frac{(LC'(j) - C_0(j)) - (LC'(j-1) - C_0(j-1))}{\tau}\right| \\
&\overset{(5.51)}{=} \frac{(LC'(j) - C_0(j)) - (LC'(j-1) - C_0(j-1))}{\tau} \\
&\overset{(5.18)}{=} \frac{(LC'(j-1) + \gamma_{j-1}\tau(1+\rho) - (C_0(j-1)+\tau)) - (LC'(j-1) - C_0(j-1))}{\tau} \\
&= \gamma_{j-1}(1+\rho) - 1 \overset{\gamma_{j-1}\leq\gamma_{\max}}{\leq} \gamma_{\max}(1+\rho) - 1 \\
&= \rho - (1-\gamma_{\max}) - (1-\gamma_{\max})\rho \quad \leq \quad \rho - (1-\gamma_{\max})
\end{aligned}$$

Für beliebiges n ergibt sich die Fehlerabschätzung als $\frac{1}{n}$ der Summe der Fehler für die Einzelintervalle, also $\leq \frac{n}{n}(\rho - (1-\gamma_{\max}))$.

(5.59) Zuerst wird der Fehler für $n = 1$ gezeigt:
Für Intervalle mit $LC'(j) - C_0(j) \geq LC'(j-1) - C_0(j-1)$ kann die obige Abschätzung aus dem Beweis für (5.58) übernommen werden:

$$\left|\frac{LC'(j) - LC'(j-1)}{\tau} - 1\right| = 1(\rho - (1-\gamma_{\max}))$$

Für Intervalle mit $LC'(j) - C_0(j) < LC'(j-1) - C_0(j-1)$ gelten folgende Implikationen:

Aus (5.53) folgt, daß dann $LC'(j-1) - C_0(j-1) > O_\infty$ sein muß.
Aus (5.54) folgt dann, daß $LC'(j-1) - C_0(j-1) - O_\infty \leq 0.08867\, O_\infty$ ist.
Aus (5.53) folgt dann, daß $|LC'(j) - C_0(j) - O_\infty| \leq 0.08867\, O_\infty$ ist. Zusammen ergibt dies, daß $|(LC'(j) - C_0(j)) - (LC'(j-1) - C_0(j-1))| \leq 2 \cdot 0.08867 \cdot O_\infty$ ist und

$$\begin{aligned}
\left|\frac{LC'(j) - LC'(j-1)}{\tau} - 1\right| &= \left|\frac{(LC'(j) - C_0(j)) - (LC'(j-1) - C_0(j-1))}{\tau}\right| \\
&\leq (2 \cdot 0.08867)\frac{O_\infty}{\tau} \quad = \quad 0.17734\frac{fO_\infty}{\tau}\frac{1}{f} \\
&\overset{f>\frac{1}{2}}{\leq} 0.17734\frac{fO_\infty}{\tau}\frac{1}{1/2} \\
&= 0.35468(\rho - (1-\gamma_{\max}) - (1-\gamma_{\max})\rho) \\
&\leq 0.35468(\rho - (1-\gamma_{\max}))
\end{aligned}$$

Für ein beliebiges Intervall ist somit der Fehler $\leq \max(1, 0.35468)(\rho - (1-\gamma_{\max})) = \rho - (1-\gamma_{\max})$. Mit der obigen Überlegung gilt diese Abschätzung dann auch für beliebiges n. Somit ist die linke Seite des Minimums in (5.59) gezeigt.
Die rechte Seite ergibt sich aus (5.54) mit folgender Abschätzung:

$$\begin{aligned}
\left|\frac{LC'(j+n) - LC'(j)}{n\tau} - 1\right| &= \left|\frac{(LC'(j+n) - C_0(j+n)) - (LC'(j) - C_0(j))}{n\tau}\right| \\
&\overset{(5.54)}{\leq} \frac{(1.08867 - 0) \cdot O_\infty}{n\tau} \quad = \quad 1.08867\frac{fO_\infty}{\tau}\frac{1}{f} \\
&\overset{f\geq\frac{1}{2}}{\leq} \frac{1.08867}{n}\frac{fO_\infty}{\tau}\frac{1}{1/2} \quad \leq \quad \frac{2.17734}{n}(\rho - (1-\gamma_{\max}))
\end{aligned}$$

(5.60) Analog zum letzten Beweis folgt aus (5.55) und $f \geq 1$

$$\begin{aligned}
\left|\frac{LC'(j+n) - LC'(j)}{n\tau} - 1\right| &\leq \frac{1.41276 - 0}{1 \cdot n}(\rho - (1-\gamma_{\max})) \\
&= \frac{1.41276}{n}(\rho - (1-\gamma_{\max}))
\end{aligned}$$

(5.60) Ebenso analog zum letzten Beweis folgt aus (5.56) und $f > 1.224635$

$$\begin{aligned}
\left|\frac{LC'(j+n) - LC'(j)}{n\tau} - 1\right| &\leq \frac{1.51338 - 0}{1.224635 \cdot n}(\rho - (1-\gamma_{\max})) \\
&\leq \frac{1.2358}{n}(\rho - (1-\gamma_{\max}))
\end{aligned}$$

(5.62) Für jedes Intervall ist der Fehler:

$$\begin{aligned}
\left|\frac{LC'(j) - LC'(j-1)}{\tau} - 1\right| &= \left|\frac{(LC'(j) - C_0(j)) - (LC'(j-1) - C_0(j-1))}{\tau}\right| \\
&\overset{(5.57)}{=} \frac{fO_\infty}{\tau} \\
&= \rho - (1-\gamma_{\max}) - (1-\gamma_{\max})\rho \leq \rho - (1-\gamma_{\max})
\end{aligned}$$

Mit der Überlegung zu (5.58) gilt diese Abschätzung dann auch für beliebiges n.

□

Satz 13 läßt sich folgendermaßen zusammenfassen:

Satz 14 *Im Szenarium 2 auf S. 51 gilt für $\gamma_{\max}(1+\rho) > 1$ folgende Fehlerabschätzung für die lokale Bestimmung des Zeitintervalls zwischen zwei Ereignissen e^j und e^l mit $j \neq l$*

$$\left| \frac{LC'(j)-LC'(l)}{C_0(j)-C_0(l)} - 1 \right| \leq 1.41276(\rho - (1-\gamma_{\max})). \tag{5.63}$$

Die folgenden Szenarien relativieren Szenarium 2 in zweierlei Hinsicht:

- Szenarium 3 verzichtet auf die symmetrische Nachrichtenübertragung, und
- Szenarium 4 verzichtet auf den konstanten zeitlichen Abstand zwischen zwei Nachrichtenübertragungen. Das Szenarium idealisiert diesen Fall in der Form, daß abwechselnd ein kurzer und ein langer Nachrichtenabstand vorhanden ist.

Szenarium 3 *Dieses Szenarium ist wie Szenarium 2 auf S. 51, aber zum Zeitpunkt $j = 0$ wird nur eine Nachricht von Prozeß 0 nach Prozeß 1 gesandt und bei allen weiteren Zeitpunkten ($j > 0$) nur in der Gegenrichtung.*

Das Verhalten von LC' ist im Szenarium 3 in den Fällen $f \in (0, 1.272858]$ des Satzes 10 identisch mit dem Verhalten im Szenarium 2, d.h. gemäß (5.34) bis (5.41). Für die Fehleranalyse sei nur der erste Zyklus betrachtet, d.h. bis LC'_1 erstmals unter C_0 fällt. Alle weiteren Zyklen sind ähnlich wie der erste Zyklus, sobald durch das schneller Gehen von C_1 auch LC'_1 wieder gegenüber C_0 vorgeht. Für die Fälle $f \in [1.272859, \infty)$ muß zwischen LC'_0 und LC'_1 unterschieden werden, da hier LC'_0 auf C_0 absinkt, während LC'_1 unter C_0, aber höchstens auf C_1 absinkt.
Es sei j der Zeitpunkt, an den LC'_1 erstmals unter C_0 fällt.

Für $f \in (1, \sqrt{2})$ gilt gemäß (5.55) und (5.56), daß

$$0 \leq LC'_1(j-1) - C_0(j-1) \leq 1.51338\, O_\infty$$

Der Fall ist in Abb. 5.8 dargestellt.

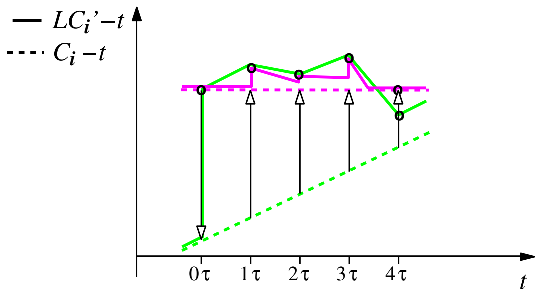


Abb. 5.8: LC'_i im Szenarium 3 für $f \in (1, \sqrt{2})$

Damit kann man den Fehler folgendermaßen abschätzen:

$$
\begin{aligned}
\left|\frac{LC'(j)-LC'(j-1)}{\tau}-1\right| &= \left|\frac{\underbrace{(LC'(j)-C_0(j))}_{<0}-\underbrace{(LC'(j-1)-C_0(j-1))}_{\geq 0}}{\tau}\right| \\
&= -\frac{(LC'(j)-C_0(j))-(LC'(j-1)-C_0(j-1))}{\tau} \\
&\overset{(5.18)}{=} -\frac{(LC'(j-1)+\gamma_{j-1}\tau(1+\rho)-(C_0(j-1)+\tau))-(LC'(j-1)-C_0(j-1))}{\tau} \\
&= 1-\gamma_{j-1}(1+\rho) \\
&\overset{(5.19)}{=} 1-\gamma_{\max}\left[1-\left(\frac{\underbrace{LC'(j-1)-C_0(j-1)}_{0\leq\ldots\leq 1.51338\,O_\infty}}{D}\right)^2\right](1+\rho) \\
&\leq 1-\gamma_{\max}\left[1-\left(\frac{1.51338\,O_\infty}{D}\right)^2\right](1+\rho) \\
&= 1-\gamma_{\max}(1+\rho)+1.51338^2\,(\rho-(1-\gamma_{\max})(1+\rho)) \\
&= -(\rho-(1-\gamma_{\max})(1+\rho))+1.51338^2\,(\rho-(1-\gamma_{\max})(1+\rho)) \\
&= 2.02676(\rho-(1-\gamma_{\max})(1+\rho)) \\
&\leq 2.03(\rho-(1-\gamma_{\max}))
\end{aligned}
$$

Für den Fall $f \geq \sqrt{2}$ besteht der erste Zyklus nur aus den beiden ersten Intervallen. Es gilt:

$$
\begin{aligned}
LC_1'(1)-C_0(1) &\overset{(5.57)}{=} fO_\infty \overset{(5.50)}{=} \rho-(1-\gamma_{\max})(1+\rho) \\
\gamma_1 &\overset{(5.19)}{=} \gamma_{\max}\left[1-\left(\frac{fO_\infty}{D}\right)^2\right] \\
LC_1'(2)-C_0(2) &= \max(LC'(1)+\gamma_1\tau(1+\rho), 2\tau(1+\rho)-D)-2\tau \\
&= \max(fO_\infty+\tau-\gamma_{\max}\left[1-\left(\frac{fO_\infty}{D}\right)^2\right]\tau(1+\rho), \\
&= \max(fO_\infty-\tau(\gamma_{\max}(1+\rho)-1)-\gamma_{\max}\tau(1+\rho)\left(\frac{fO_\infty}{D}\right)^2, 2\tau\rho-D) \\
&\overset{(5.50)}{=} \max(-\gamma_{\max}\tau(1+\rho)\left(\frac{fO_\infty}{D}\right)^2, 2\tau\rho-D) \\
&\overset{(5.50)}{=} \max(-\tau^3\frac{\gamma_{\max}(1+\rho)[\rho-(1-\gamma_{\max})(1+\rho)]^2}{D^2}, 2\tau\rho-D)
\end{aligned}
$$

Für die folgende Fallunterscheidung sei vorausgesetzt, daß $\sqrt[3]{\rho} \ll 1$. Die Fälle sind in Abb. 5.9 dargestellt.

1. Wenn $\tau \lesssim \frac{D}{\sqrt[3]{\gamma_{\max}(1+\rho)[\rho-(1-\gamma_{\max})(1+\rho)]^2}} \simeq D\,\rho^{-2/3}$, d.h. wenn $f \lesssim \sqrt[6]{1/\rho}$, dann ist die linke Seite der Maximumbildung maßgebend und

$$\begin{aligned}\left|\frac{LC'(j)-LC'(j-1)}{\tau}-1\right| &= -\frac{(LC_1'(2)-C_0(2))-(LC_1'(1)-C_0(1))}{\tau} \\ &= \rho-(1-\gamma_{\max})(1+\rho)+\tau^2\frac{\gamma_{\max}(1+\rho)[\rho-(1-\gamma_{\max})(1+\rho)]^2}{D^2} \\ &\leq \rho-(1-\gamma_{\max})(1+\rho)+\sqrt[3]{\gamma_{\max}(1+\rho)[\rho-(1-\gamma_{\max})(1+\rho)]^2} \\ &\simeq [\rho-(1-\gamma_{\max})]+\sqrt[3]{[\rho-(1-\gamma_{\max})]^2}\end{aligned}$$

2. Wenn $\tau > \frac{D}{\sqrt[3]{\gamma_{\max}(1+\rho)[\rho-(1-\gamma_{\max})(1+\rho)]^2}} \simeq D\,\rho^{-2/3}$, d.h. wenn $f \gtrapprox \sqrt[6]{1/\rho}$, dann ist die rechte Seite der Maximumbildung maßgebend und

$$\begin{aligned}\left|\frac{LC'(j)-LC'(j-1)}{\tau}-1\right| &= -\frac{(LC_1'(2)-C_0(2))-(LC_1'(1)-C_0(1))}{\tau} \\ &\leq \rho-(1-\gamma_{\max})(1+\rho)\frac{D}{\tau} \\ &\leq \rho-(1-\gamma_{\max})(1+\rho)+\sqrt[3]{\gamma_{\max}(1+\rho)[\rho-(1-\gamma_{\max})(1+\rho)]^2} \\ &\simeq [\rho-(1-\gamma_{\max})]+\sqrt[3]{[\rho-(1-\gamma_{\max})]^2}\end{aligned}$$

Zusammenfassend gilt dann:

$$\left|\frac{LC'(j)-LC'(l)}{C_0(j)-C_0(l)}-1\right| \leq \begin{cases} 2.03[\rho-(1-\gamma_{\max})] & \text{für } f<\sqrt{2} \\ [\rho-(1-\gamma_{\max})]+\sqrt[3]{[\rho-(1-\gamma_{\max})]^2}) & \text{für } f\geq\sqrt{2}\end{cases}$$

für f definiert gemäß (5.28) und (5.29).

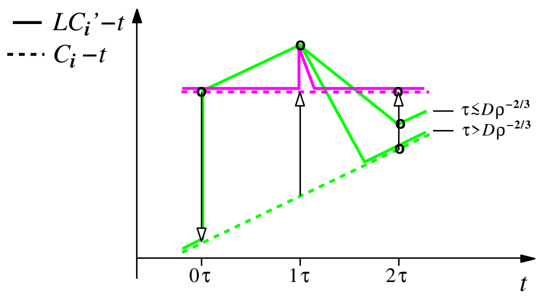


Abb. 5.9: LC_i' im Szenarium 3 für $f \geq \sqrt{2}$

Szenarium 4 *Dieses Szenarium ist wie Szenarium 3 auf S. 62, die Intervalle sind jedoch unterschiedlich lang:* $C_0(j)-C_0(j-1)=\tau_j$ *und* $C_1(j)-C_1(j-1)=\tau_j(1+\rho)$. *Die Intervallänge variiert um höchstens* $\kappa=\tau_{\max}/\tau_{\min}$ *mit* $\tau_{\max}=\max_{j=1,2,\ldots}\tau_j$ *und* $\tau_{\min}=\min_{j=1,2,\ldots}\tau_j$.

Die Fehlerabschätzung zu diesem Szenarium ist identisch bis auf die Fälle, in denen LC' auf C absinkt, d.h. für $f \geq 1.272859$. Da bei der Fehlerabschätzung zum Szenarium 3 für $f<\sqrt{2}$ die Maximum-Bildung mit C_1 nicht berücksichtigt werden mußte, ändert sich in diesem Fall die Fehlerabschätzung auch nicht. Das gleiche gilt für $f \geq \sqrt{2}$ und $f \leq \sqrt[6]{1/\rho}$. Nur für $f \geq \sqrt{2}$ und $f \geq \sqrt[6]{1/\rho}$ kann durch ein kurz nach dem zweiten Ereignis folgendes drittes Ereignis ein um κ größerer Fehler auftreten, da nun die Kurve um den Faktor κ steiler sein kann als im Szenarium 2, bevor der rechte Teil in der Maximumbildung in (5.18) relevant wird. Zusammenfassend gilt dann:

Satz 15 *Im Szenarium 4 gilt für $\gamma_{\max}(1+\rho) > 1$ folgende Fehlerabschätzung näherungsweise für die lokale Bestimmung des Zeitintervalls zwischen zwei Ereignissen e^j und e^l mit $j \neq l$*

$$\left|\frac{LC'(j)-LC'(l)}{C_0(j)-C_0(l)}-1\right| \tag{5.64}$$

$$\leq \begin{cases} 2.03[\rho-(1-\gamma_{\max})] & \textit{für } f_{\max} < \sqrt{2} \\ \max\Big(2.03[\rho-(1-\gamma_{\max})],\ [\rho-(1-\gamma_{\max})]+\sqrt[3]{[\rho-(1-\gamma_{\max})]^2}\Big) & \textit{für } f_{\max} \geq \sqrt{2} \textit{ und } f_{\max} \leq \sqrt[6]{1/\rho} \\ \kappa \max\Big(2.03[\rho-(1-\gamma_{\max})],\ [\rho-(1-\gamma_{\max})]+\sqrt[3]{[\rho-(1-\gamma_{\max})]^2}\Big) & \textit{für } f_{\max} \geq \sqrt{2} \textit{ und } f_{\max} > \sqrt[6]{1/\rho} \end{cases}$$

Beispiel 3: Für $1-\gamma_{\max} = 2\cdot 10^{-5}$, $\rho = 10^{-4}$ und $\kappa = 30$ ergibt dies gemäß Satz 15

$$\begin{aligned} \left|\tfrac{LC'(j)-LC'(l)}{C_0(j)-C_0(l)}-1\right| \quad \langle^7 &\leq 30\cdot\max\Big(2.03\,[8\cdot 10^{-5}],\ [8\cdot 10^{-5}]+\sqrt[3]{800000}\cdot 10^{-5}\Big) \\ &\leq 30\cdot\max\,(2.03\cdot 8,\ 8+92.84)\,10^{-5} \qquad \rangle^7 \\ &\leq 0.031 \end{aligned}$$

Abb. 5.10 zeigt den Verlauf von $LC'-C_0$ über t/D mit $t = \tau j$ in den Fällen des Szenariums 2 und den Verlauf von $LC_1'-C_0$ in den Fällen des Szenariums 3. Die Abbildung zeigt, daß die ansteigenden Teilstücke niemals steiler als die einheitliche Anfangssteigung sind und daß die abfallenden Teilstücke nur unwesentlich steiler als die negative Anfangssteigung sind, mit der Ausnahme von $f = 2$ im Szenarium 3.

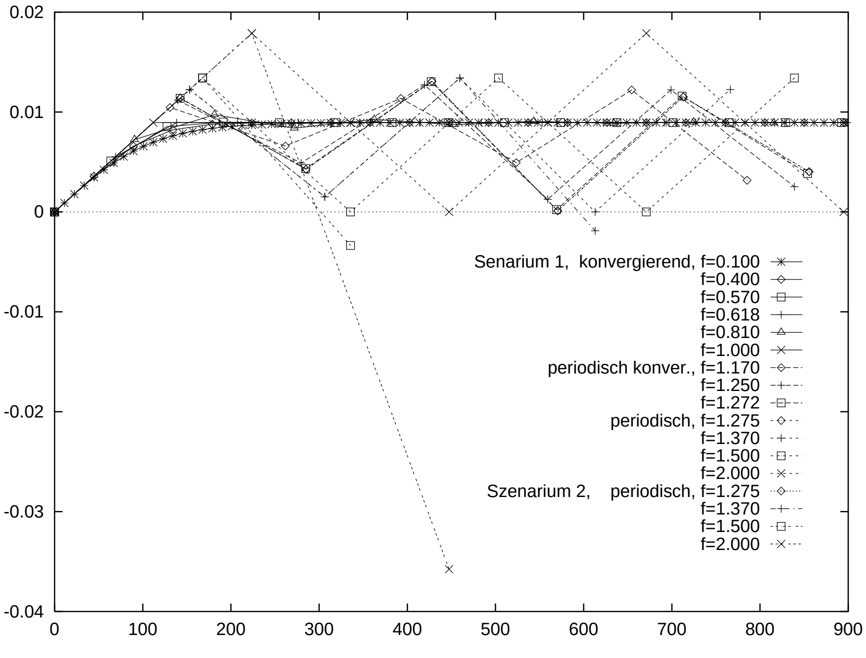


Abb. 5.10: $(LC'-C_0)/D$ über t/D im Beispiel 3

Die Abschätzung zum Szenarium 3 ist auch gültig, wenn das dritte Ereignis in Prozeß 1 ein lokales Ereignis ist. Sie wird im Gegensatz zu den Abschätzungen zum Regler $\mathcal{A}$ durch

[7] siehe Fußnote 1 auf Seite 48

die rückwärtige Amortisation nicht verbessert. Da das Verhältnis zwischen den Abständen von KommunikationsEreignissen und den Abständen lokaler Ereignisse im Bereich $\kappa > 10^4$ liegen kann, können im Beispiel 3 dann auch Fehler größer als 10 % auftreten. Dies wird aber durch den Regler $\mathcal{D}$ verhindert, der den Fehler auf $\leq 1 - \gamma_{\mathcal{D}}$ begrenzt. Zusammenfassend gilt dann:

Satz 16 *Im Szenarium 4 beschränken die Regler $\mathcal{B}$ und $\mathcal{D}$ für $\gamma_{\max}(1+\rho) > 1$ den relativen Fehler auf*

$$\begin{aligned}
&\left|\tfrac{LC'(j)-LC'(l)}{C_0(j)-C_0(l)} - 1\right| \\
&\leq \min\left(1-\gamma_{\mathcal{D}},\ \kappa \max\left(2.03[\rho-(1-\gamma_{\max})],\ [\rho-(1-\gamma_{\max})] + \sqrt[3]{[\rho-(1-\gamma_{\max})]^2}\right)\right) &(5.65)\\
&\leq 1-\gamma_{\mathcal{D}} &(5.66)
\end{aligned}$$

Die gleiche Abschätzung gilt auch für LC_0^A. Da das Szenarium 4 den generellsten Fall darstellt, gilt obige Abschätzung generell für Fehler innerhalb eines Prozesses, die bei Kombination von Regler $\mathcal{B}$ und $\mathcal{D}$ entstehen.

5.4 Charakteristik der Regler $\mathcal{C}$ und $\mathcal{D}$

Im vorhergehenden Kapitel wurde gezeigt, daß die Regelung zwar zu einem Schwingen der geregelten logischen Uhr führen kann, aber daß dieses Schwingen für den Fehler bei der Messung lokaler Zeitintervalle keinen wesentlichen Einfluß besitzt, da der Fehler im wesentlichen durch die Konstanten $\gamma_{\mathcal{D}}$ in (5.66) bestimmt ist.

Auf eine schärfere Abschätzung analog zu (5.65) wird hier verzichtet, da normalerweise nur der Regler $\mathcal{B}$ zur Anwendung kommt.

Da die Gleichung (5.66) unabhängig von der Art der Regelung des Reglers $\mathcal{B}$ ist, sondern nur auf der Begrenzung der γ-Werte durch Regler $\mathcal{D}$ beruht, kann sie auch für die Kombination von Regler $\mathcal{C}$ und $\mathcal{D}$ übernommen werden.

5.5 Zusammenfassung der Fehleranalyse

Wie schon bei der Analyse des Reglers $\mathcal{C}$ wird auch hier auf eine scharfe Abschätzung wie in (5.65) verzichtet. Die wesentlichen Aussagen ergeben sich aus den Gleichungen (5.9) und (5.12) für die rückwärtige Amortisation und der Gleichung (5.66) für die Regelung.

Satz 17 *Es sei $\rho_{\max}$ eine obere Grenze der absoluten Gangabweichungen der Prozessoruhren, d.h.*

$$\underset{k, e_k^j \in E_k, e_k^l \in E_k}{\forall} \left| \frac{C_k(t(e_k^l)) - C_k(t(e_k^j))}{t(e_k^l) - t(e_k^j)} - 1 \right| \leq \rho_{\max} \tag{5.67}$$

Diese Definition berücksichtigt, daß die Prozessoruhren nur für die aufgezeichneten Ereignisse einen Wert besitzen müssen. Es seien $\bar{A}_i$ und $\check{A}_i$ obere Schranken für den Amortisationsgradient, definiert durch

$$\bar{A}_i = \max\{A_{\text{def}},\ \epsilon \tilde{\kappa}_{a,i} \tfrac{1}{2}(1 + \tilde{\kappa}_{b,i})\}$$

mit

$$\tilde{\kappa}_{a,i} = \frac{\tilde{\mu}}{\min\limits_{\substack{j<l,\ e_i^j \ Sendeereignis, \\ e_i^l \ Empfangsereignis,\ m(e_i^l) \le (1+\epsilon/2)\tilde{\mu}}} (t(e_i^l) - t(e_i^j))}$$

$$\tilde{\kappa}_{b,i} = \frac{\max\limits_{\substack{j<l,\ e_i^j, e_i^l \ aufeinanderfolgende \\ Empfangsereignisse\ mit\ m(e) \le (1+\epsilon/2)\tilde{\mu}}} (t(e_i^l) - t(e_i^j))}{\tilde{\mu}} \frac{(1-\gamma_{\max}) + \rho_{\max}\gamma_{\max}}{2\epsilon}$$

und

$$\check{A}_i = \max\{A_{\mathrm{def}},\ \check{\kappa}_i\}$$

mit

$$\check{\kappa}_i = \frac{\max_{m,k,t}(C_m(t) - C_k(t))}{\min_{j<l,\ e_i^j \ Sendeereignis,\ e_i^l \ Empfangsereignis}(t(e_i^l) - t(e_i^j))}$$

Die Konstanten $\gamma_{\max}$, $\gamma_{\mathcal{D}}$ und A_{def} seien folgendermaßen gewählt:
$1 - \gamma_{\max}$ sei positiv und größer als die für das System im allgemeinen anzunehmende Gangabweichung der Prozessoruhren, z.B. $\gamma_{\max} := 1 - 2 \cdot 10^{-4}$.
$1 - \gamma_{\mathcal{D}}$ sei positiv und größer als die für das System im Extremfall anzunehmende Gangabweichung, gemessen über größere Zeiträume, d.h. es darf sein, daß z.B. bei einem spontanen Uhrensprung eine noch wesentlich größere Gangabweichung auftritt. Beispiel: $\gamma_{\mathcal{D}} := 1 - 5\%$.
A_{def} wähle man gemäß einer gemünschten optimalen Genauigkeit, z.B. $A_{\mathrm{def}} := 0.5\%$.
Dann gilt für die rückwärtig amortisierte geregelte logische Uhr

$$\begin{aligned}
\epsilon_{R,I,S}^{LC_i^A} = \max_j \left| \frac{LC_i'(e_i^{j+1}) - LC_i'(e_i^j)}{t(e_i^{j+1}) - t(e_i^j)} - 1 \right| &\le max(\bar{A}_i, \check{A}_i)(1+\rho_{\max}) + (1-\gamma_{\max}) + \rho_{\max} \\
&\quad + (1-\gamma_{\mathcal{D}}) + \rho_{\max} + (1-\gamma_{\mathcal{D}})\rho_{\max} \\
&= max(\bar{A}_i, \check{A}_i) + (1-\gamma_{\max}) + (1-\gamma_{\mathcal{D}}) + 2\rho_{\max} \\
&\quad + (max(\bar{A}_i, \check{A}_i) + 1 - \gamma_{\mathcal{D}})\rho_{\max} \qquad (5.68)
\end{aligned}$$

Das letzte Produkt ist hierbei ein quadratischer Term, der in der Regel vernachlässigt werden kann.

Beweis: Folgt direkt aus den Prämissen und (5.9), (5.12), (5.66), (5.16) und (5.17) unter Berücksichtigung, daß

$$\frac{\Delta LC}{\Delta t} - 1 = (\frac{\Delta LC}{\Delta C} - 1)(\frac{\Delta C}{\Delta t} - 1) + (\frac{\Delta LC}{\Delta C} - 1) + (\frac{\Delta C}{\Delta t} - 1)$$

und, daß die Fehler durch die Regelungen $\mathcal{B}$ und $\mathcal{C}$ wahlweise entstehen und daher nicht addiert werden müssen.

□

Die Fehler-Abschätzung in Satz 17 gibt eine obere Grenze für den maximalen Fehler bei der Messung von Zeitintervallen innerhalb eines Anwendungsprozesses an. In der Praxis liegt der reale maximale Fehler normalerweise weit darunter, da meist $\check{A}$ und $\bar{A}$ durch ein kleines Vielfaches von A_{def} ersetzt werden können. Ebenso wird normalerweise Regler $\mathcal{B}$ statt Regler $\mathcal{C}$ aktiv. Dadurch kann man den Term $(1-\gamma_{\mathcal{D}})$ durch das Minimum aus (5.65) ersetzen, wobei in der Regel sich κ und der potenzierte Term in (5.65) nur schwach auswirken, und somit der maximale Fehler in der Praxis meist ein Vielfaches von

$$\max((1-\gamma_{\max}),\, \rho_{\max},\, A_{\text{def}}) \tag{5.69}$$

ist. Der durchschnittliche Fehler, der zu dem Fehler der Prozessoruhren durch die geregelte logische Uhr hinzugefügt wird, ist meistens nur ein Bruchteil von (5.69). Damit liegt bei entsprechender Wahl von $\gamma_{\max}$ und A_{def} der Fehler durchschnittlich im Bereich der Ganggenauigkeit der ursprünglichen Prozessoruhren, also von $\rho_{\max}$.

Bei einer Wahl von $A_{\text{def}} = 0.5\,\%$ ist somit in der Regel die Gangabweichnung auf ein kleines Vielfaches von $0.5\,\%$ begrenzt und somit in der Regel kleiner als $5\,\%$. Die Fehleranalyse in diesem Kapitel betrachtete die maximale Gangabweichung, da für diesen Wert eine akzeptable Obergrenze wichtig ist. Dies wurde schon eingangs in der Anforderung (A1) auf Seite 16 festgestellt. In der Praxis liegt die durchschnittliche Gangabweichung[4] weit unterhalb dieser Obergrenze. In dem im nachfolgenden Kapitel auf Seite 72, unten links, dargestellten Beispiel beträgt die maximale Gangabweichung $1.137\,\%$, während die durchschnittliche Gangabweichung nur $0.004\,\%$ beträgt.

Die Problematik der absoluten Synchronisationsfehler betrifft im wesentlichen die Messung der Nachrichtenlaufzeiten d_M. Die geregelte logische Uhr garantiert hierzu nur die Gültigkeit der Uhrenbedingung, bzw. daß grundsätzlich $d_M \geq \mu$ gilt. In der Praxis ist trotzdem eine gute Synchronisation zu erwarten, wenn durch Nachrichten mit kurzen Nachrichtenlaufzeiten die geregelte logische Uhr sich mit dem Maximum der Prozessoruhren synchronisiert. Eine weitere Qualitätssteigerung kann erreicht werden, wenn man der geregelten logischen Uhr eine konstante Uhrenkorrektur vorschaltet, indem vor dem Start der parallelen Anwendung eine präzise Synchronisation durchgeführt wird. Bei Offline-Monitoren kann die Präzision der Messungen von Nachrichtenlaufzeiten nochmals verbessert werden, indem, statt der konstanten, eine lineare Uhrenkorrektur vorgeschaltet wird, wenn also – auf der Basis von präzisen Synchronisationen vor dem Start und nach Beendigung der parallelen Anwendung – über die Programmlaufzeit die in [MT95] beschriebene, linear interpolierte Korrektur durchgeführt wird, und anschließend die geregelte logische Uhr mit rückwärtiger Amortisation angewandt wird.

[4]Die durchschnittliche Gangabweichung sei definiert als der Durchschnitt der Gangabweichungen in allen Zeitintervallen, die durch zwei aufeinanderfolgende Ereignisse innerhalb eines Prozesses begrenzt sind.

Kapitel 6

Implementierung und Test

6.1 Geregelte logische Uhr *ohne* rückwärtige Amortisation

Abb. 6.1 zeigt das Schema der Implementierung der geregelten logischen Uhr als Filter. In einem einzigen Durchgang **(single pass)** wird ein Tracefile mit möglicherweise fehlerhaften Zeitstempeln gelesen, die geregelte logische Uhr berechnet, und die Ereignisse mit den damit gebildeten neuen Zeitstempeln wieder ausgegeben. Es wird vorausgesetzt, daß die Ereignisse in der Eingabedatei gemäß ihren Zeitstempeln sortiert sind. Der *Eingabe-Puffer* dient zur Speicherung von Empfangsereignissen und weiteren ggf. nachfolgenden Ereignissen, solange die geregelte logische Uhr des zugehörigen Sendeereignisses noch nicht berechnet ist, und diese Ereignisse folglich noch nicht bearbeitet sind. Der *Puffer für berechnete Sendeereignisse* speichert die neuen Zeitstempel von Sendeereignissen, bis das zugehörige Empfangsereignis bearbeitet wird.

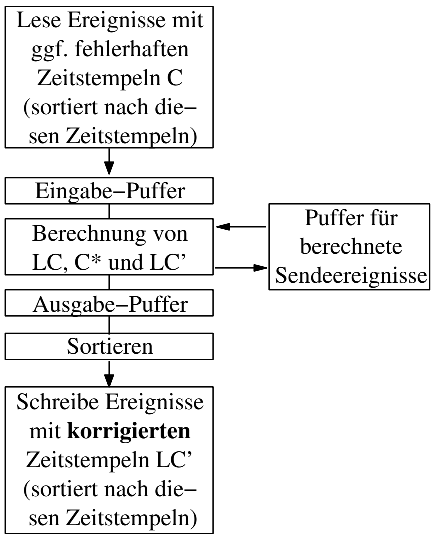


Abb. 6.1: Implementierung als Filter

Der *Ausgabe-Puffer* speichert die Ereignisse mit den neu berechneten Zeitstempeln, solange es möglich ist, daß noch weitere Ereignisse mit geringeren Zeitstempeln kommen können. Der Ausgabe-Puffer ist zweigeteilt. In die erste Gruppe werden alle Ereignisse eingespeichert, deren neuer Zeitstempel kleinergleich als eine Schranke ist, in die zweite Gruppe alle restlichen. Die erste Gruppe wird sortiert, ausgegeben, und anschließend die zweite Gruppe zur ersten gemacht und die Schranke neu auf das Maximum aller neuen Zeitstempel in dieser Gruppe festgelegt, sobald ein Ereignis eingelesen und entweder bearbeitet oder im Eingabe-Puffer zwischengespeichert wurde, und dessen alter Zeitstempel größergleich der Schranke zwischen den beiden Ausgabegruppen ist.

Die benötigte Rechenzeit ist linear in der Anzahl der Ereignisse, da die Anzahl der bei der Ausgabe zu sortierenden Ereignisse durch $(3+1) \cdot \max_{i,k}(|C_i - C_k|) \cdot \Delta E/\Delta t$ beschränkt ist. $\Delta E/\Delta t$ gibt hierbei die Anzahl der Ereignisse pro Zeitintervall an. Der Faktor 3 ergibt sich aus der oberen Grenze in der Reglercharakteristik des hinreichenden Reglers ($\mathcal{C}$).

Der Sortiervorgang entfällt, wenn die Ereignisse in der Ausgabedatei nur bezüglich des jeweiligen Prozesses sortiert sein müssen. Alternativ zu dem hier angegebenen Sortiervorgang kann auch der im nachfolgenden Kapitel angegebene Misch-Algorithmus gewählt werden. Der Sortiervorgang hat den Vorteil, daß sein Aufwand gegen Null geht, wenn die ursprünglichen Zeitstempel schon korrekt waren, bzw. wenn nur minimale Korrekturen nötig sind.

6.2 Implementierung der rückwärtigen Amortisation

Zur Realisierung der rückwärtigen Amortisation muß zwischen die geregelte logische Uhr und den Ausgabe-Puffer für jeden Prozeß eine zweigeteilte First-In-First-Out-Queue eingeführt werden, s. Abb. 6.2. Solange zu Sendeereignissen noch das zugehörige Empfangsereignis fehlt, werden diese und nachfolgende weitere Ereignisse in dem Verzögerungspuffer gehalten. Dann gelangen die Ereignisse in den Amortisationspuffer. Für jedes neu hinzugekommene Empfangsereignis wird, wenn nötig, der Amortisationsalgorithmus ausgeführt. Das letzte Ereignis am Ausgang des Amortisationspuffers wird in den Misch-Algorithmus (multiway merge) übergeben, sobald es einen älteren Zeitstempel besitzt als das Ereignis am Eingang des Amortisationspuffers abzüglich der Länge des Amortisations-Intervalls. Die Länge des Amortisationsintervalls ist durch den Uhrenunterschied dividiert durch den maximal gewünschten Fehler gegeben. Der Anwender sollte eine realistische obere Grenze für den Uhrenunterschied vorgeben. Zusätzlich wird das Amortisationsintervall verlängert sobald eine größere Uhrendifferenz zwischen zwei Prozessen ermittelt wird. Eine zu gering angegebene Grenze erhöht den Amortisations-Fehler bei Ereignissen, deren Sprunghöhe größer ist als der bis dahin vorgegebene oder ermittelte maximale Uhrenunterschied. Der maximale Uhrenunterschied wird in diesem Fall angepaßt. Der maximal gewünschte Amortisationsfehler ist frei wählbar. Die Implementierung wählt als Voreinstellung 0,5 %. Die Länge des Amortisationsintervalls bestimmt den notwendigen Speicherbedarf. Der Misch-Algorithmus muß die schon

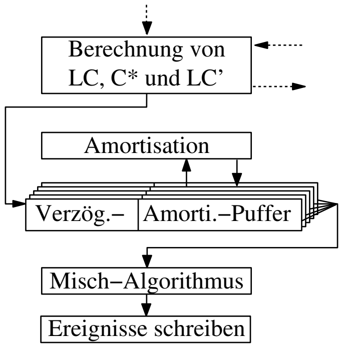


Abb. 6.2: Implementierung der rückwärtigen Amortisation

Gesamt-ausführungs-zeit [sec]	Anteil Prog. laden, Trace lesen u. schreiben [sec]	Anteil Berechnung der gereg. log. Uhr [sec]	Prozessor	Takt-rate [MHz]	Beschaffungs-zeitpunkt des Rechners
18.6	5.7	12.1	MIPS R3000	33	6/1992
3.1	1.2	1.9	MIPS R8000	75	11/1994
0.9	0.4	0.5	MIPS R10000	195	12/1997
3.4	2.9	0.5	wie vorstehend, aber Tracedatenzugriff via NFS und Ethernet		

Tabelle 6.1: Ausführungszeiten für einen 1.9 MB großen Tracefile

sortierten Ergebnisse der einzelnen Prozesse in einen global geordneten Datenstrom mischen. Bei Implementierung mittels eines vollständigen Binärbaums ist der Aufwand proportional zu $\log_2(Anzahl\ der\ Prozesse)$.

Es ist nicht zu empfehlen, den im vorigen Kapitel angegebenen Sortieralgorithmus zu verwenden, da die im Vergleich zur maximalen Uhrendifferenz große Länge des Amortisationsintervalls die Längen der auftretenden Sortierintervalle stark vergrößern. Daher ist in der Regel der zu $\log_2(Anzahl\ der\ Prozesse)$ proportionale Mischaufwand geringer als der Sortieraufwand, wenn man den Algorithmus aus dem vorherigen Kapitel übernehmen würde.

Tabelle 6.1 zeigt die Ausführungszeiten für einen 1.9 MB großen binären VAMPIR-Tracefile [NA95, NAW+96] mit von 16 Prozessen erzeugten 56622 Ereignissen. Für die zweite Spalte wurde eine Version mit entferntem Berechnungsalgorithmus eingesetzt. Die dritte Spalte wurde nicht direkt gemessen, sondern ergibt sich aus der Differenz der ersten beiden. Die ersten 4 Zeilen wurden mit lokalen Platten ermittelt. Die Tabelle gibt den Hinweis, daß langfristig die überproportionale Steigerung der Rechnergeschwindigkeiten gegenüber den Plattengeschwindigkeiten dazu führt, daß der Hauptaufwand im Lesen und Schreiben der Tracefiles liegen wird. Die letzte Zeile zeigt diesen Effekt schon heute, wenn man einen langsameren Plattenzugriff zugrunde legen muß, wie z.B., wenn auf die Daten über ein lokales Netz zugegriffen werden muß.

Da es sich bei der geregelten logischen Uhr um einen Filteralgorithmus handelt, kann er direkt in das Visualisierungswerkzeug eines Monitors integriert werden. Dann entfällt der Lese- und Schreibaufwand vollständig. Dies ist ein wesentlicher Vorteil gegenüber allen Algorithmen, die zwei Durchläufe benötigen, wie z.B. die in Kap. 3.3 dargestellten Verfahren von Duda et al. und Hofmann.

6.3 Test des Algorithmus

Der Algorithmus wurde mit drei Beispielen getestet:

1. Mittels eines Tracefiles einer realen Anwendung mit 16 Prozessen auf einem Parallelrechner mit ungenügender Uhrensynchronisation.

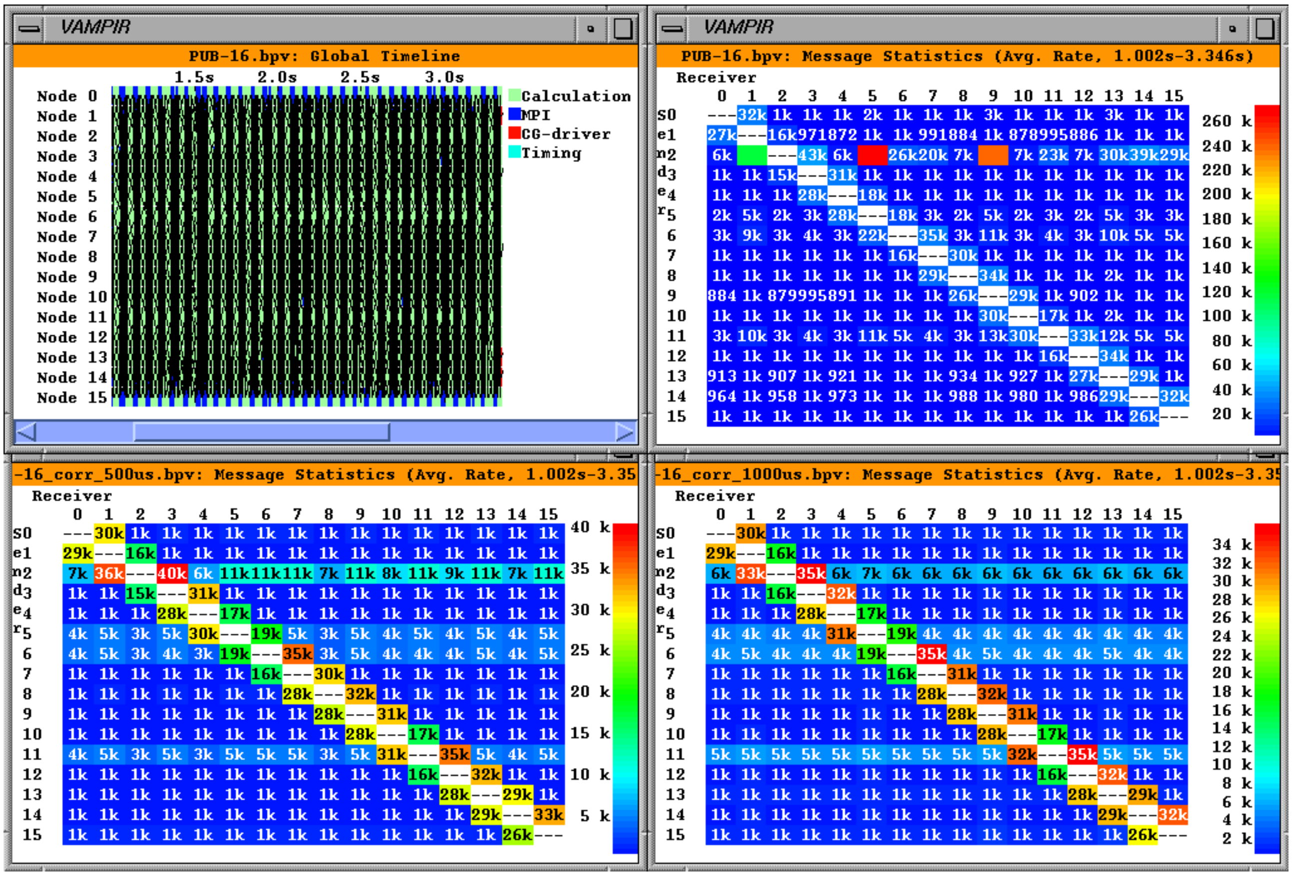
2. Ein Beispiel mit großem Uhrentick. Es handelt sich hierbei um das Beispiel 1 in [Rab96]. Ohne die Regelung, d.h. bei $\gamma \equiv 1$, würde bei diesem Beispiel LC' unbegrenzt wachsen.

3. Mittels einer simulierten FE-Rechnung.

Diese sind in den folgenden Kapiteln beschrieben.

6.3.1 Tracefile einer realen Anwendung

Abb. 6.3 auf S. 72 zeigt links oben einen Ausschnitt des Zeitliniendiagramms einer realen Anwendung und rechts oben das zugehörige Diagramm der durchschnittlichen Transferraten. Beide Diagramme sind auf der Basis der nicht korrigierten Zeitstempel der Prozessoruhren erstellt. Man sieht bei von Knoten 2 ausgehenden Nachrichten extrem unrealistische Transferraten im Bereich 120 (nach Knoten 1), 270 (nach Knoten 5) und 240 (nach Knoten 9) kB/sec. Nach einer Korrektur mittels der geregelten logischen Uhr mit $\mu = 500\mu$s ergibt sich die wesentlich realistischere Darstellung links unten. Hierbei war nur der Regler $\mathcal{A}$ mit




```
> pvcorrect -maxerr 0.1 -mmd 500 PUB-16.bpv

Differences new timestamps - old timestamps at the last event in [ms]
Node  i=      0      1      2      3      4      5      6      7
    0+i  0.0000 1.1534 0.1186 0.3070 0.1886 1.1617 0.4447 0.4085
    8+i  0.2006 1.2684 0.0000 0.4334 0.0000 1.2413 0.8874 0.4823

Maximal clock differences = 1302.92 us,  used start-value = 1000.00 us

ADVICE -- use for correcting similar tracefiles:  -cldiff 1303

Minimal difference between the original timestamps of two events
in the same process = 5.000 us

Minimal message delay between 2 processes =
   (min=620.200us,  avg=2093.456us,  max=3157.700us)
with min/avg/max over all paires of processes with messages in both
directions.  These values include an error <= 2*(max. clock drifts
over the whole time)

The correction of the timestamps causes additional errors on the length
of time intervals between two successive events in the same process:
 13689 intervals with        error  =   0%
 42617 intervals with    0% < error <= 0.1% : avg.= 0.002%, max= 0.003%
   300 intervals with 0.1% < error          : avg.= 0.468%, max= 1.137%
Error summary:
 56606 intervals with 0.0% <=error <= 1.137% : avg.= 0.004%
```

```
> pvcorrect -maxerr 0.1 -mmd 1000 PUB-16.bpv

Differences new timestamps - old timestamps at the last event in [ms]
Node  i=      0      1      2      3      4      5      6      7
    0+i  0.0152 1.4885 0.0000 0.6455 0.1850 1.5002 0.7830 0.7470
    8+i  0.2067 1.6069 0.0766 0.7774 0.1430 1.5825 0.9634 0.8395

Maximal clock differences = 1858.71 us,  used start-value = 1000.00 us

ADVICE -- use for correcting similar tracefiles:  -cldiff 1859

Minimal difference between the original timestamps of two events
in the same process = 5.000 us

Minimal message delay between 2 processes =
   (min=620.200us,  avg=2093.456us,  max=3157.700us)
with min/avg/max over all paires of processes with messages in both
directions.  These values include an error <= 2*(max. clock drifts
over the whole time)

ADVICE -- use for correcting similar tracefiles:  -mmd 496.160
but this value can be meaningless if the clock drifts are to large.

The correction of the timestamps causes additional errors on the length
of time intervals between two successive events in the same process:
  5394 intervals with        error  =   0%
 49484 intervals with    0% < error <= 0.1% : avg.= 0.002%, max= 0.077%
  1728 intervals with 0.1% < error          : avg.= 0.982%, max= 8.522%
Error summary:
 56606 intervals with 0.0% <=error <= 8.522% : avg.= 0.032%
```

Abb. 6.3: Zeitliniendiagramm und durchschnittliche Transferraten bei einer realen Anwendung, Transferraten rechts oben basierend auf C, links unten basierend auf LC', berechnet mit $\mu = 500\mu$s, rechts unten basierend auf LC', berechnet mit $\mu = 1000\mu$s, jeweils darunter mit Fehleranalyseprotokoll des geregelten logischen Uhrenprogramms

$1-\gamma = 2\cdot10^{-5}$ aktiv. Die Regler $\mathcal{B}$ und $\mathcal{C}$ stellten keinen Regelbedarf fest. Die Amortisationsparameter waren 1000 μs als erwartete maximale Uhrendifferenz und 0.1 % als gewünschter maximaler Amortisationsfehler. Die zur linken unteren Abbildung gehörende, von der Implementierung der geregelten logischen Uhr ausgegebene Analyse zeigt, daß für Zeitintervalle zwischen zwei aufeinanderfolgenden Ereignissen im selben Prozeß der maximale Fehler 1.137 % und der durchschnittliche Fehler $4 \cdot 10^{-5}$ beträgt. Eine detaillierte Analyse zeigt, daß nur bei zwei Prozessen die in Abb. 4.12 auf S. 41 dargestellte Amortisation stückweise linear ist und hierbei in den steileren Bereichen einen Fehler größer als $4 \cdot 10^{-5}$ erzeugt. Hierbei sind zusammen aber nur ca. 0.26 % der gesamten Zeitspanne des Traces (16×5.34 sec) betroffen.

Die Qualität der **Verbesserung** von Zeitmessungen der Nachrichtenlaufzeiten ergibt sich aus dem Vergleich der beiden Diagramme rechts oben und links unten. Die wesentlichen Fehler in der oberen Darstellung sind eliminiert. Die verbleibenden Abweichungen in den Zeilen 2, 5, 6 und 11 können real sein, aber auch auf mangelnder Synchronisation, da Nachrichten mit minimaler Laufzeit fehlten.

In dem Diagramm rechts unten wurde bei der geregelten logischen Uhr ein $\mu = 1000\mu$s zugrunde gelegt. Dieses μ liegt deutlich über dem mit horizontalen Parallelstreifen in Hofmanns Algorithmus (s. Abb. 3.3 auf S. 18) ermittelten minimalen Nachrichtenlaufzeiten von 620μs, aber noch knapp unter dem Wert, ab welchem die logische Uhr mit $\gamma \equiv 1$ immer stärker vorgeht. Der Algorithmus arbeitet hierdurch sehr scharf und erzielt bzgl. Zeitmessung von Nachrichtenlaufzeiten $d_M^{LC'}$ ein noch besseres Ergebnis, wie man dem Diagramm entnehmen kann, doch Zeitmessungen innerhalb eines Prozesses werden deutlich schlechter. Der Fehler beträgt hier maximal 8.522 % und durchschnittlich nur 0.032 %.

Wählt man dagegen die Voreinstellung von $\mu = 250\mu$s, dann ist für Zeitintervalle zwischen zwei aufeinanderfolgenden Ereignissen im selben Prozeß der maximale Fehler nur $3\cdot10^{-5}$ und der durchschnittliche ungefähr $1 \cdot 10^{-5}$.

6.3.2 Ein Beispiel mit großem Uhrentick

Dieses Beispiel testet die Regler und die Qualität der neuen Zeitstempel in einer extremen Umgebung: Einerseits besitzen die Prozessoruhren einen Uhrentick der 40 mal größer als die minimale Nachrichtenlaufzeit ist, und andererseits sind die Uhren so justiert, daß ohne die Regelung das LC' gegenüber C unbeschränkt wachsen würde. 20 Knoten (0...19) senden eine Nachrichtenkette von Knoten 0, über Knoten 1, 2, usw. nach Knoten 19 und anschließend wieder zurück nach Knoten 0. Die Uhren haben einen Uhrentick von 10 ms, die Nachrichtenlaufzeit sei 250 μs, und das Weiterschalten der Uhren sei so, daß in der Nachrichtenkette von Knoten 0 nach 19 die Nachrichten jeweils eine scheinbare Nachrichtenlaufzeit von -9.750 ms besitzen und in der Nachrichtenkette von 19 nach 0 eine scheinbare Laufzeit von -250 ms. Abb. 6.4 auf S. 74 zeigt oben den Anfang des zugehörigen Zeitliniendiagramms auf der Basis der Zeitstempel der Prozessoruhren. Es stellt den Sachverhalt extrem verzerrt dar. Der wirkliche Verlauf ist nicht erkennbar. Bei Knoten 0, z.B., wird die zweite Nachricht nach Knoten 1 bei 20.25 ms abgeschickt (s. vertikaler Pfeil). Die Darstellung dieser Nachricht ist revers, d.h. es hat den Anschein, daß sie von Knoten 1 nach Knoten 0 geht, bzw. die dargestellte Laufzeit ist negativ (-9.7 ms, s. innere Box). Um ca. 9.9 ms später wird dann von Knoten 0 als nächstes eine Nachricht von Knoten 1 empfangen. Aufgrund des bis dahin noch nicht erfolgten Uhrenticks schließt sich diese in der Darstellung – fehlerhafterweise – sofort an, s. horizontaler Pfeil. Sie ist ebenfalls revers dargestellt.

Die Korrektur der Zeitstempel mittels der geregelten logischen Uhr ermöglicht die sehr

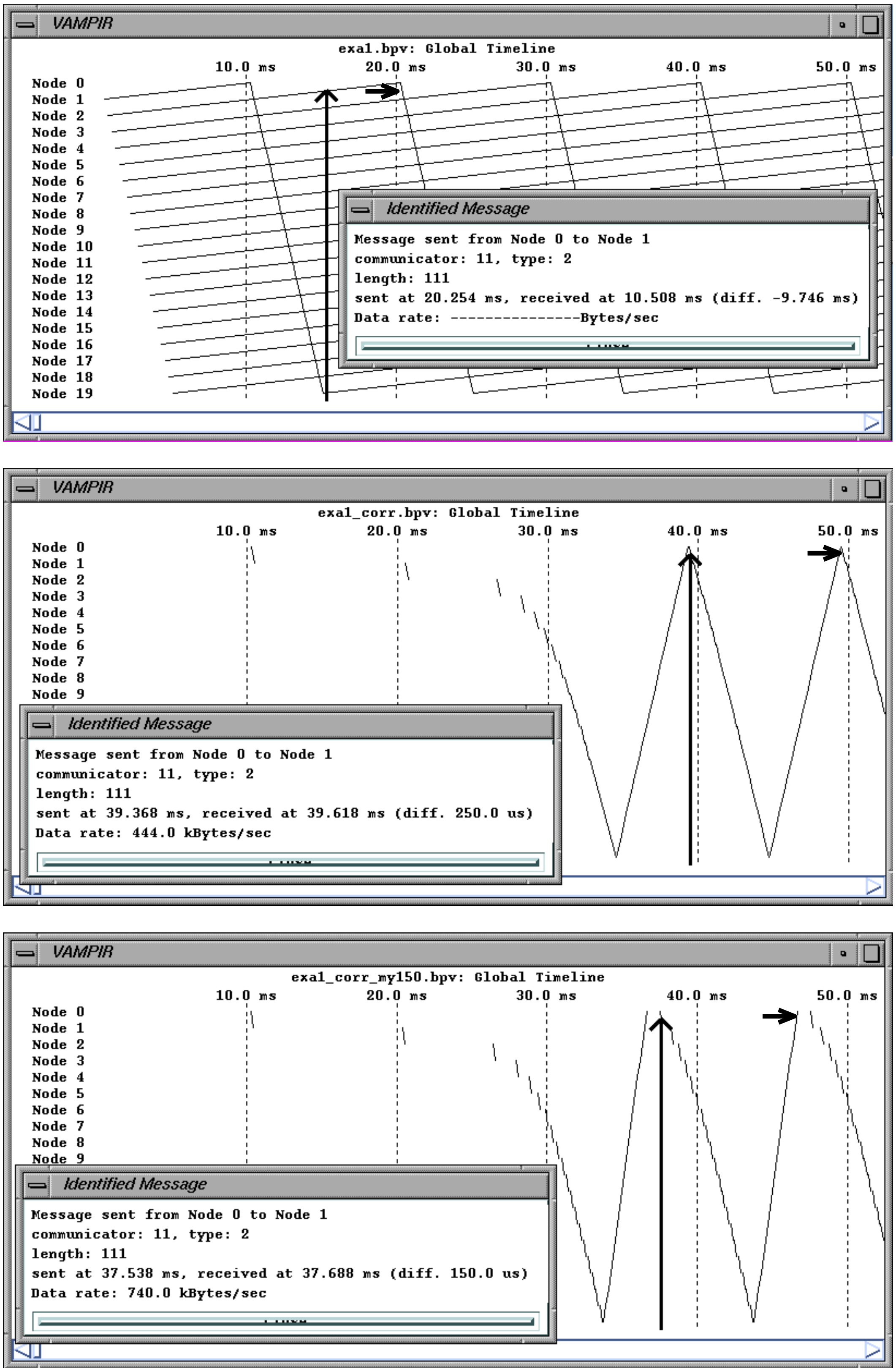


Abb. 6.4: Beispiel mit großem Uhrentick

realistische Darstellung des Programmverlaufs in dem mittleren Diagramm. Hierbei wurde bei der Korrektur $\mu = 250\,\mu$s angenommen. Links sieht man den Einschwingvorgang während der ersten 7 Nachrichten. Die beiden oben beschriebenen Nachrichten findet man in dieser Darstellung nun richtig von Knoten 0 nach Knoten 1 (s. vertikaler Pfeil, 39.368-39.618 ms) und von Knoten 1 nach Knoten 0 richtigerweise ca. 9.5 ms später (s. horizontaler Pfeil, 49.036-49.286 ms). Eine Analyse während der Berechnung der neuen Zeitstempel zeigt, daß die Regler γ auf ca. 0.17 herabregeln.

Die untere Abbildung zeigt eine weniger korrekte Darstellung. Hierbei wurde für die Berechnung der geregelten logischen Uhr ein $\mu = 150\,\mu$s angenommen, d.h. die reale Nachrichtenlaufzeit lag $100\,\mu$s über der zugrunde gelegten minimalen Nachrichtenlaufzeit. Dieses Beispiel zeigt, daß die geregelte logische Uhr auch bei extrem nicht-kontinuierlichen Prozessoruhren und auch bei extremem Regelbedarf eine sinnvolle Darstellung des Programmflusses ermöglichen kann. Dies war durch die in Kap. 3.3 beschriebenen Methoden seither nicht möglich.

6.3.3 Simulation einer FE-Rechnung

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 logische Uhr errechnet zu $C(t(e_i^j))$ die Werte $LC'(e_i^j)$ und $LC(e_i^j)$,
- und ein Analysemodul bewertet die geregelte logische Uhr.

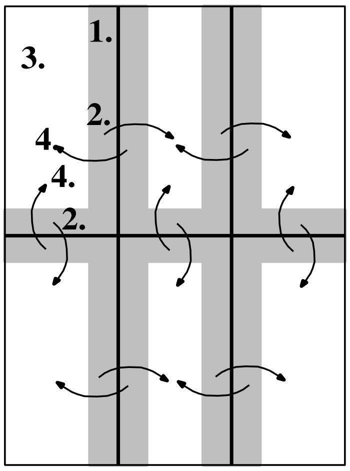


Abb. 6.5: Eine fiktive parallele FE-Berechnung

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:

1. Berechnung der finiten Elemente an den Rändern zu den Nachbarn.
2. Versenden dieser neuen Ergebnisse zu den Nachbarn.
3. Berechnen der restlichen finiten Elemente.
4. Empfangen der neuen Nachbarwerte.

Diese Simulationsumgebung ermöglichte das Testen und Verifizieren der theoretischen Grundlagen der geregelten logischen Uhr und ist in [Rab96] genauer beschrieben.

Kapitel 7

Bewertung

Zuerst werden in den Kapiteln 7.1 bis 7.3 die Einsatzmöglichkeiten der geregelten logischen Uhr erörtert. In Kapitel 7.4 wird ein extremer Anwendungsfall mit nicht ausreichend synchronisierten Uhren bei einem Kommunikationsnetz mit sehr geringer Nachrichtenlaufzeit vorgestellt. In Kapitel 7.5 werden dann die existierenden Verfahren zur nachträglichen Uhrenkorrektur nochmals aufgezeigt und das seither fortschrittlichste Verfahren, das von Hofmann und Hilgers detailliert vorgestellt. In Kapitel 7.6 werden die Grenzen dieser existierenden verfahren aufgezeigt. Im wesentlichen sind diese Verfahren nicht anwendbar, wenn die Uhrengeschwindigkeiten der Prozessoruhren nicht konstant sind. In Kapitel 7.7 entwickle ich einen neuen Algorithmus zur nachträglichen Synchronisation von Prozessoruhren mit variablen Uhrengeschwindigkeiten auf der Basis des Verfahrens von Hofmann und Hilgers. Anschließend werden in Kapitel 7.8 die existierenden Verfahren, dieser neue Algorithmus für variable Uhrengeschwindigkeiten und die geregelte logische Uhr mit rückwärtiger Amortisation miteinander verglichen und bewertet.

7.1 Zur Klassifikation der Anwendungen

In Kapitel 3.1 auf Seite 15 wurden die Anwendungsklassen vorgestellt und deren Anforderungsprofil in Kap. 3.2 aufgezeigt.

Die geregelte logische Uhr ist für Monitoringwerkzeuge konzipiert. Beim *Monitoring* verteilter und paralleler Anwendungen kann sie als Filter eingesetzt werden. Ein Tracefile mit Rückwärtsbezügen, d.h. dessen Zeitstempel die Uhrenbedingung (Gl. 2.4) nicht erfüllen, wird nachträglich so modifiziert, daß er einerseits die Uhrenbedingung erfüllt, und daß andererseits die Zeitstempel für Zeitmessungen möglichst nur geringfügig verändert werden.

Hierbei sei zwischen *online* und *offline* Werkzeuge unterschieden. *Online* Monitorwerkzeuge ermöglichen eine Betrachtung der Ereignisse, während die Anwendung noch läuft, die die Ereignisse erzeugt. *Offline* Monitorwerkzeuge dienen zur nachträglichen Analyse eines Tracefiles, nachdem die Anwendung vollständig abgeschlossen ist.

Die geregelte logische Uhr mit rückwärtiger Amortisation ist ein Filter, das eine kontinuierliche Bearbeitung der eingehenden Tracedaten durchführt und hierzu einen Teil der Daten puffert. Die Pufferung ist einerseits zur Berechnung der logical Clock nötig, bei der ein Empfangsereignis erst bearbeitet werden kann, wenn das zugehörige (in physikalischer Zeit auch vorhergehende) Sendeereignis schon bearbeitet ist. Andererseits benötigt auch die Amortisation einen Puffer. Hierbei kann ein Ereignis den Puffer erst verlassen, wenn es aus dem

Amortisationsintervall herauskommt. In der in Kap. 6.2 auf S. 70 vorgestellten Implementierung endet das Amortisationsintervall mit dem neuesten bearbeiteten Empfangsereignis, bei dem zu allen vorhergehenden Sendeereignissen im gleichen Prozeß die zugehörigen Empfangsereignisse schon einen logischen Zeitstempel besitzen.

7.2 Nachträgliche Tracebearbeitung

Da die geregelte logische Uhr zuerst nur für eine Bearbeitung der Ereignisse vom Beginn an entworfen ist, gibt es beim Einsatz innerhalb eines Monitorwerkzeuges Probleme, wenn eine beliebige Positionierung innerhalb des Tracefiles mit anschließendem Vorwärts- oder Rückwärtsscrollen möglich sein soll. Einerseits kann man die Korrekturen von Anfang des Tracefiles an berechnen und für das Scrollen innerhalb des Files zwischenspeichern, andererseits kann man auch die geregelte logische Uhr 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 Sende- und Empfangsereignissen vertauscht. Letzteres Verfahren hat den Nachteil, daß die durch die geregelte logische Uhr modifizierten Zeitstempel von der Wahl des Anfangsereignisses abhängen, d. h. der Anwender sieht das gleiche Ereignis u. U. mit unterschiedlichen Zeitstempeln dargestellt.

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

- die Zeit der lokalen Prozessoruhr, ggf. linear korrigiert mittels der in [MT95] beschriebenen SBA-Methode (**s**ampling **b**efore and **a**fter); sie kann für Zeitdifferenzmessungen innerhalb eines Prozesses und zum Referenzieren eingesetzt werden;

- die Zeit der geregelten logischen Uhr; sie dient zur Darstellung der Ereignisse in Zeitdiagrammen und für Zeitdifferenzmessungen zwischen Ereignissen in verschiedenen Prozessen; ihr Wert ist jeweils abhängig vom Anfangsereignis nach der letzten Repositionierung.

Prinzipiell sei empfohlen, die geregelte logische Uhr zusammen mit einer expliziten Uhrensynchronisation am Anfang und am Ende der zu beobachtenden Anwendung einzusetzen. Dies reduziert den Korrekturaufwand, so daß die geregelte logische Uhr nur noch Probleme aufgrund der nicht konstanten Uhrengeschwindigkeiten ausgleichen muß.

Es wird empfohlen, eine Abschätzung für die minimale Nachrichtenlaufzeit μ als Nebenprodukt des eingesetzten Synchronisationsalgorithmus zu erlangen, oder eine Hardware bezogene Abschätzung durchzuführen. Diese Abschätzung sollte vor dem ersten Ereignis in den Tracefile eingetragen werden. Die Streuung der Nachrichtenlaufzeiten kann noch einen signifikanten Bereich unterhalb der Laufzeiten mit der größten Häufigkeit besitzen. Unsere Messungen haben in einem Ethernet und in einem FDDI-Ring ergeben, daß die minimale Laufzeit im allgemeinen größer als ein Viertel der häufigsten Roundtriptime eines leeren RPCs ist. Da aber Nachrichten mit minimalen Nachrichtenlaufzeiten bei einer derartigen Wahrscheinlichkeitsverteilung äußerst selten sind, kann man in der Regel $\mu = 90\% \cdot \frac{1}{2} \textit{häufigste Roundtriptime}$ wählen.

Der Algorithmus der geregelten logischen Uhr ist unempfindlich gegen eine zu große Wahl von μ. Dies führt im wesentlichen nur dazu, daß die Regler γ stärker herunterregeln müssen und daß damit größere Gangabweichungen verbunden sind. Andererseits führt dies in der

Regel zu geringeren Synchronisationsfehlern, da meist die Nachrichten eine größere Nachrichtenlaufzeit als die für das Netzwerk gültige minimale Nachrichtenlaufzeit besitzen. Eine gute Balance von Gangabweichungen und Synchronisationsfehlern kann meist erreicht werden, wenn man μ etwa 20 % größer als die minimale Nachrichtenlaufzeit wählt. Ein Beispiel hierzu findet man rechts unten in Abb. 6.3 auf Seite 72.

7.3 Online-Tracebearbeitung

Die geregelte logische Uhr kann auch online, wie z.B. für Monitoring-Komponenten innerhalb von zentralen *Debuggingwerkzeugen für parallele Programme* eingesetzt werden. Die fehlende Synchronisation am Ende sollte hierbei durch eine ressourcenarme fortlaufende Synchronisation parallel zur Anwendung ersetzt werden, wie z. B. mit xntp [Mil92]. Die Synchronisation am Anfang kann dann in der Regel ebenfalls entfallen.

Ein online Einsatz ist möglich, da die Pufferung zeitlich begrenzt werden kann. Die Pufferung hat, wie am Ende von Kap. 7.1 beschrieben, zwei Aspekte. Für die weitere Analyse sei vorausgesetzt, daß die Übermittlung der Ereignisse zur geregelten logischen Uhr ohne relevante Verzögerung stattfindet. Irrelevant ist hierbei die Pufferung eines Empfangsereignisses bis das zugehörigen Sendeereignis vorhanden ist, da das Sendeereignis physikalisch vor dem Empfangsereignis auftreten muß. Die Pufferung innerhalb des Amortisationsintervalls muß gegenüber der in Kapitel 6.2 angegebenen Implementierung aber abgeändert werden. Zum einen kann die Einschränkung, daß bei einem zu bearbeitenden Empfangsereignis e_R zu den vorangegangenen Sendeereignissen e_s schon ihre zugehörigen Empfangsereignisse e_r vorhanden sein müssen, derart reduziert werden, daß auf ein solches Empfangsereignis e_r nur bis zum Zeitpunkt $C(t(e_R)) + \mu + \max_{i,k,t} |C_i(t) - C_k(t)|$ zuzüglich der maximal möglichen Verzögerung τ_{transfer} bei der Übermittlung der Ereignisse zur geregelten logischen Uhr gewartet werden muß. Außerdem kann das Amortisationsintervall auch unabhängig von zu amortisierenden Empfangsereignissen e_R weitergeschoben werden. Dadurch kann jedes Ereignis e den Amortisationspuffer verlassen, wenn

$$\begin{aligned} LC'(e) < C_{\text{Monitor}} &- \max_{i,t}(C_i(t) - C_{\text{Monitor}}) \\ &- \mu - \max_{i,k,t} |C_i(t) - C_k(t)| \\ &- A_{\text{def}} \max(\Delta C_{\text{init}}, \max_{i,k,t} |C_i(t) - C_k(t)|) - \tau_{\text{transfer}} \end{aligned} \tag{7.1}$$

Sowohl die Abweichung zwischen der Uhr des Monitors C_{Monitor} und denen der zu monitorenden Prozesse C_i, als auch die zwischen den Uhren der Anwendungsprozesse sollten gemessen werden. Das Maximum sollte dann mit einem Sicherheitsfaktor abgeschätzt werden, der die Auswirkung der Gangabweichung während der Laufzeit der Anwendung ausreichend berücksichtigt. Die letzten 4 Summanden stellen die reale Verzögerung des Filters der geregelten logische Uhr mit rückwärtiger Amortisation dar. Wenn durch eine ressourcenarme Synchronisation die gegenseitige Abweichung der Prozessoruhren C_i auf unter 5 ms gehalten und dann $C_{\text{init}} = 5$ ms gesetzt wird, und ein 2-prozentiger Fehler bei der Amortisation akzeptiert werden kann, d.h. $A_{\text{def}} = 0.05$ ist, und wenn die minimale Nachrichtenlaufzeit μ unter 10 ms liegt, und die Transferzeit τ_{transfer} kürzer als 100 ms ist, dann beträgt die gesamte Verzögerung nur 215 ms. Eine derartige Verzögerung ist bei einem Monitorwerkzeug noch akzeptabel.

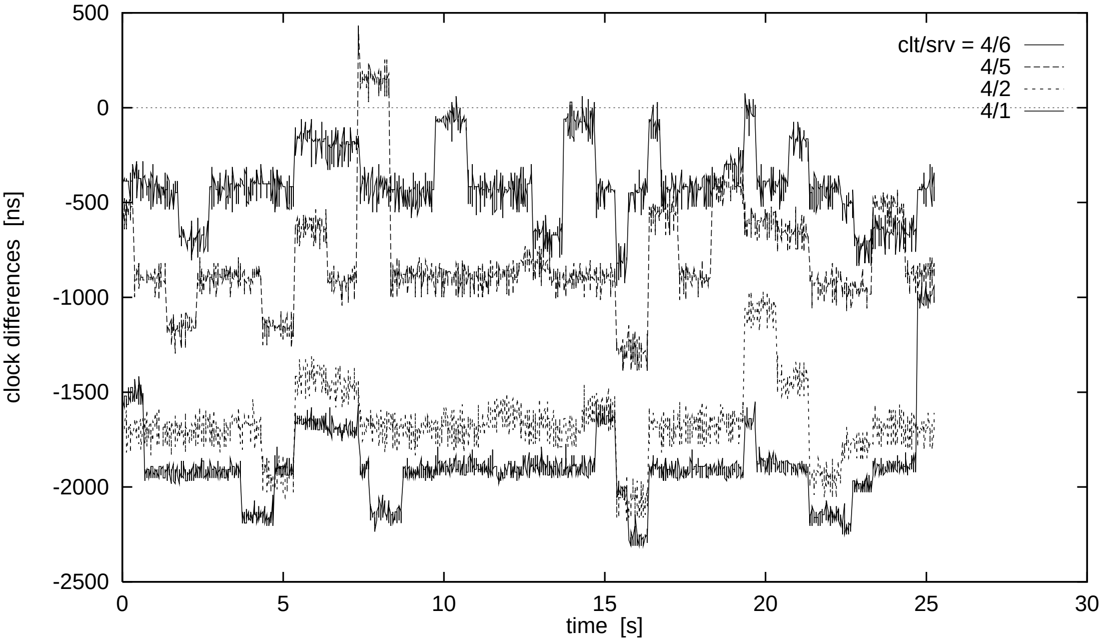


Abb. 7.1: Uhrendifferenzen auf der CRAY T3E

Im Gegensatz zu dem in [Rab96, Rab97] beschriebenen Regler handelt es sich hier bei den Reglern $\mathcal{B}$ und $\mathcal{C}$ um Algorithmen, die eine zentrale Verfügbarkeit aller Zeitstempel erfordern, und die daher nicht für eine auf die einzelnen Prozesse verteilte Implementierung geeignet sind. Daher ist das Einsatzgebiet der hier beschriebenen Regler auf Monitorwerkzeuge und vergleichbare Anwendungen beschränkt.

7.4 Art der physikalischen Zeitmessung

Die geregelte logische Uhr ist konzeptionell unabhängig von der Art der Zeitmessung innerhalb eines parallelen Systems. Sie kann z.B. eingesetzt werden, wenn die Zeitmessung C_i direkt mit physikalischen Quarzuhren realisiert ist. Hierbei gleicht sie die Fehler durch die Gangabweichung der Uhren und die fehlende Synchronisation aus.

Die Uhren können aber auch schon synchronisiert sein. Der Einsatz der geregelten logischen Uhr ist dann sinnvoll, wenn die Synchronisation nicht ausreicht, d.h. wenn Uhrendifferenzen auftreten können, die größer als die minimale Nachrichtenlaufzeit sind. Abb. 7.1 auf S. 80 zeigt eine Messung auf der CRAY T3E/512-900 des Rechenzentrums der Universität Stuttgart am 29. April 1998. Die Messung wurde mit einer Partition von 8 Prozessoren durchgeführt. Es sind die Uhrendifferenzen der Uhren der Prozesse 6, 5, 2 und 1 zur Uhr des Prozesses 4 (jeweils logische Nummern innerhalb der Partition) dargestellt. Die Uhrendifferenzen wurden mittels zweier aufeinanderfolgender Nachrichten vom *client* Knoten 4 zu einem der *server* Knoten und zurück ermittelt. Hierbei wurden nur Nachrichtenpaare berücksichtigt, deren Roundtriptime[1] höchstens 150 ns über der minimalen Roundtriptime lagen. Die Nachrichtenübertragung wurde mit `shmem_put` durchgeführt. Die Nachrichten-

[1] Summe der Nachrichtenlaufzeiten der beiden Nachrichten

laufzeit (halbe Roundtriptime) lag bei ca. 1150 ns. Die Abbildung zeigt, daß die Uhren im Sekundentakt mittels abruptem Vor- und Zurücksetzen synchronisiert werden und hierbei bezüglich der Uhrendifferenzen Sprünge im Bereich von 1000 ns auftreten können. Da außerdem die Uhrendifferenzen im Bereich bis 2300 ns liegen und das Betriebssystem jederzeit Partitionen verschieben kann, sind zudem Sprünge in diesem Bereich beim Monitoring von parallelen Anwendungen zu berücksichtigen. Dies heißt, daß auf der T3E bei shmem-basierten Programmen die Nachrichtenlaufzeiten geringer sein können als die Uhrenfehler. Die geregelte logische Uhr kann derartige Fehler dann ausgleichen. Die geregelte logische Uhr kann aber auch eingesetzt werden, wenn die Uhren mittels eines linearen Ausgleichs synchronisiert werden, d.h. das Vor- und Zurücksetzen über einen Zeitraum linear verteilt wird.

Beim Einsatz der geregelten logischen Uhr ist hierbei zu beachten, daß derartige Synchronisationsverfahren die maximale Gangabweichung der Uhren C_i ändert. Dies ist bei der Wahl der Werte für $\gamma_{\mathcal{A}}$, $\gamma_{\mathcal{D}}$ und A_{def} zu berücksichtigen.

7.5 Existierende Verfahren zur nachträglichen Uhrenkorrektur

Wie schon in Kap. 3.3 aufgezeigt, ist die nachträgliche Bestimmung der Uhrenabweichungen eine Alternative zur geregelten logischen Uhr. Im wesentlichen werden die in Kap. 3.3 angegebenen Verfahren von Duda et al. [DHHB87], Jézéquel [Jez89a], Hofmann [Hof93a, Hof93b], Hofmann und Hilgers [HH98, Hil96] und Babaoğlu und Drummond [BD87, DB93] mit der geregelten logischen Uhr verglichen. Diese Verfahren unterscheiden sich bezüglich der Anwendbarkeit in folgenden Aspekten:

- Werden bestimmte Kommunikationstopologien vorausgesetzt?

  Babaoğlu und Drummond setzen regelmäßig wiederkehrende vollständige Kommunikationsbeziehungen voraus, während Duda et al., Jézéquel, sowie Hofmann und Hilgers keine Einschränkungen machen.

- Wird die Uhrenbedingung nach Anwendung der Uhrenkorrektur erfüllt?

  Der Algorithmus von Babaoğlu und Drummond hat dies im Gegensatz zu den anderen nicht zum Ziel. Er wird daher im folgenden nicht mehr erörtert.

- Müssen die Uhrengeschwindigkeiten konstant sein?
  – und –
  Werden Uhrenkorrekturen für eine beliebige Anzahl von Prozessoren ermittelt?

  Das Verfahren mit Bildung einer konvexen Hülle von Duda et al. beherrscht ein beliebiges Verhalten der Uhrengeschwindigkeiten beim 2-Prozeß-Problem, d.h. daß Traces von parallelen Anwendungen mit jeweils 2 Prozessen korrigiert werden können. Die übrigen Algorithmen setzen hinreichend konstante Uhrengeschwindigkeiten voraus, da bei ihnen eine lineare Approximation durchgeführt wird. Sie lösen aber dann das N-Prozeß-Problem.

Intern unterscheiden sich die Verfahren folgendermaßen:

- Das Verfahren mit der konvexen Hülle von Duda et al. (s. Abb. 3.2 auf S. 18) ermittelt stückweise lineare obere (und untere) Grenzen für die **Uhrendifferenzen zwischen**

je zwei Prozessen für das Gesamtintervall, in dem Ereignisse erzeugt wurden, wenn zwischen den Prozessen in der einen (und in der anderen) Richtung Nachrichten ausgetauscht worden sind.

Anschließend wird eine **über das Gesamtintervall lineare** obere (und untere) Grenze für die Uhrendifferenzen gebildet.

Mit einem **Ausgleichsalgorithmus** werden nun für die einzelnen Uhren lineare Korrekturfunktionen gebildet. Der Algorithmus garantiert, daß die Differenz aus den Korrekturfunktionen zu zwei Uhren innerhalb der oben ermittelten Grenzen für die Uhrendifferenzen liegen.

- Jézéquel gibt hierzu einen neuen Ausgleichsalgorithmus an.
- Das Verfahren von Hofmann und Hilgers vereinfacht sowohl den Algorithmus zur Ermittlung der Uhrendifferenzen, als auch den Ausgleichsalgorithmus.

Damit im nachfolgenden Kapitel die Grenzen des Verfahrens aufgezeigt werden können, sei hier das Verfahren von Hofmann und Hilgers gemäß [HH98] kurz zusammengefaßt:

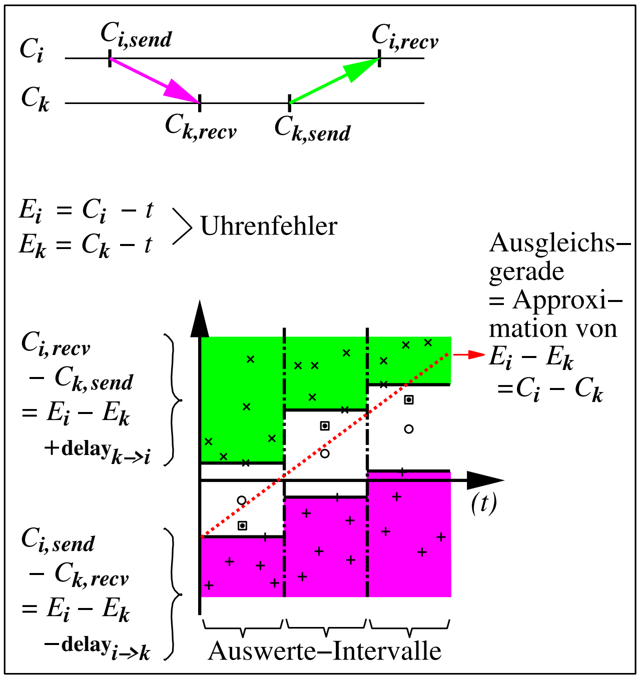


Abb. 7.2: Algorithmus von Hofmann und Hilgers

In Abb. 7.2 ist das Verfahren skizziert. Links oben sind die Bezeichnung der Zeitstempel von Sende- und Empfangsereignissen in den Prozessen i und k mit den Prozessoruhren C_i und C_k dargestellt. Das Diagramm in der Abbildung wird für jedes Prozeßpaar aufgestellt. Zuerst wird das zu untersuchende Gesamtintervall in gleichlange Auswerte-Intervalle aufgeteilt. In jedem Auswerte-Intervall werden die Werte

$$O_{ik}^{+} = C_i(e_i^{send}) - C_k(e_k^{recv}) \tag{7.2}$$

$$O_{ik}^{-} = C_i(e_i^{recv}) - C_k(e_k^{send}) \tag{7.3}$$

gebildet. Die O_{ik}^{+} und O_{ik}^{-} sind in dem Diagramm in der unteren, bzw. oberen unterlegten Fläche mit Kreuzen der Form (+) und (×) markiert. Anschließend werden hierzu die Maximas und Minimas gebildet:

$$\overline{O_{ik}} = \max_{(e_i^{send}, e_k^{recv}) \in M} (O_{ik}^{+}) \tag{7.4}$$

$$\underline{O_{ik}} = \min_{(e_k^{send}, e_i^{recv}) \in M} (O_{ik}^{-}) \tag{7.5}$$

$\overline{O_{ik}}$ ist in dem Diagramm als horizontale, obere Grenze der Fläche mit den (+)-Kreuzen in den einzelnen Auswerte-Intervallen dargestellt, $\underline{O_{ik}}$ als untere Grenze der oberen Flächen dargestellt.

Bei dem ursprünglichen Algorithmus von Hofmann wurde nun in jedem Auswerte-Intervall der Mittelpunkt der freien Fläche bestimmt (diese sind als runde Kreise dargestellt) und anschließend eine Ausgleichsgerade zu diesen Punkten ermittelt, welche dann die Werte der Uhrenkorrektur bei einem Prozessor eines Prozessorpaars angab. Dies ist in Abb. 3.3 auf S. 18 skizziert.

Die Aufgabe des folgenden Algorithmus besteht darin, Uhrenkorrekturfunktionen $Corr_i(t)$ so zu bestimmen, daß die korrigierten Zeitstempel $C_i^{corr}(t) := C_i(t) + Corr_i(t)$ die Uhrenbedingung erfüllen, wenn man das Diagramm statt für C nun für C^{corr} aufzeichnet, d.h. daß dann alle (×)-Kreuze sich oberhalb der t-Achse und alle (+)-Kreuze sich unterhalb befinden.

Bei Hofmann und Hilgers wird hierzu zuerst ein Ausgleichsalgorithmus angewandt, der zu den Wertepaaren $(\overline{O_{ik}}, \underline{O_{ik}})_{i=1..n,\, k=1..n,\, i \neq k}$ einen Satz von Uhrenkorrekturwerten $(Corr_i)_{i=1..n}$ ermittelt. Diese Werte $(Corr_i)_{i=1..n}$ werden jeweils auf der t-Achse der Mitte der Auswerte-Intervalle zugeordnet. In dem Diagramm ist jeweils der Wert $Corr_k - Corr_i$ mit einem von einem Quadrat umgebenen Punkt (⊡) dargestellt. Anschließend wird dann für jedes i eine Ausgleichsgerade durch die Punktmenge $\{(Corr_i, t_{\text{Mitte}})_{\text{alle Auswerte-Intervalle}}\}$ gelegt. Dies ergibt für jeden Index i eine lineare Uhrenkorrekturfunktion $Corr_i(t)$. Die Differenzfunktion $Corr_k(t) - Corr_i(t)$ ist in der Abbildung gestrichelt eingezeichnet. Das Anwenden der Uhrenkorrekturfunktionen führt dazu, daß in dem Diagramm die gestrichelte Ausgleichsgerade vertikal auf die t-Achse geschert wird. Dies bedeutet, daß alle Kreuze oberhalb, bzw. unterhalb der Ausgleichsgeraden nun oberhalb, bzw. unterhalb der t-Achse liegen.

Der Algorithmus zur Bestimmung der Uhrenkorrekturwerte $(Corr_i)_{i=1..n}$ entspricht einer graphentheoretischen Bestimmung der kürzesten Wege. Numerisch läßt er sich folgendermaßen ausdrücken:

```
do
  for i = 1 to n
    for² k = 1 to n
      for z = 1 to n
        O_ik = min(O_ik, O_iz + O_zk)      (untere Grenzen)
        O_ik = max(O_ik, O_iz + O_zk)      (obere Grenzen)
while (change in any O_ik + O_ik)
```

$$\underline{O_{ik}} = \min(\underline{O_{ik}}, \underline{O_{iz}} + \underline{O_{zk}})$$
$$\overline{O_{ik}} = \max(\overline{O_{ik}}, \overline{O_{iz}} + \overline{O_{zk}})$$

while (change in any $\underline{O_{ik}} + \overline{O_{ik}}$)

Mit diesem Algorithmus wurden die Werte $\underline{O_{ik}}$ und $\overline{O_{ik}}$ einerseits so verändert, daß sie noch in dem freien Bereich zwischen den unterlegten Flächen liegen und, daß es andererseits möglich ist, einen beliebigen Index i_{ref} zu nehmen und die Mittelwerte $Corr_k := (\underline{O_{ik}} + \overline{O_{ik}})/2$ als Uhrenkorrekturwerte im jeweiligen Auswerte-Intervall zu verwenden. Letzteres bedeutet, daß die Minimum- und Maximum-Bildung im vorstehenden Algorithmus garantiert, daß bei beliebiger Wahl von i_{ref} die in unserem Diagramm mit dem ⊡-Symbol dargestellten Werte $Corr_k - Corr_i$ innerhalb des weißen Streifens liegen. Diese Aussage wurde von Hofmann und Hilgers in [HH98] und [Hil96] bewiesen. Hofmann und Hilgers wählen als Index i_{ref} den Index, für den die Summe $\Sigma_{k=1}^{n} E_{i_{ref}k}$ mit $E_{ik} := (\underline{O_{ik}} - \overline{O_{ik}})/2$ minimal ist.

7.6 Grenzen der existierenden Verfahren

Hierbei muß bei allen Verfahren grundsätzlich sowohl bei der Linearisierung, als auch bei der Ausgleichsbildung kontrolliert werden, ob die Lösung noch innerhalb der durch die Nachrichtenlaufzeiten vorgegebenen Grenzen liegt. Eine der wesentlichen Problematiken ist schon in Abb. 7.2 auf Seite 82 sichtbar: Auf der rechten Seite sieht man, daßi sich die Ausgleichsgerade nicht mehr in dem weißen Mittelstreifen befindet. Wären die Zeitstempel einer Nachricht von Prozeß k nach Prozeß i so, daß ihr zugehöriges ($\times$)-Kreuz sich in dem durch die Ausgleichsgerade nach oben begrenzten Dreiecks des unterlegten Gebietes der ($\times$)-Kreuze befinden würde, dann wäre nach Anwendung der Uhrenkorrektur $C_i^{corr}(e_i^{recv}) - C_k^{corr}(e_k^{send})$ negativ, d.h. die Uhrenbedingung wäre nicht erfüllt.

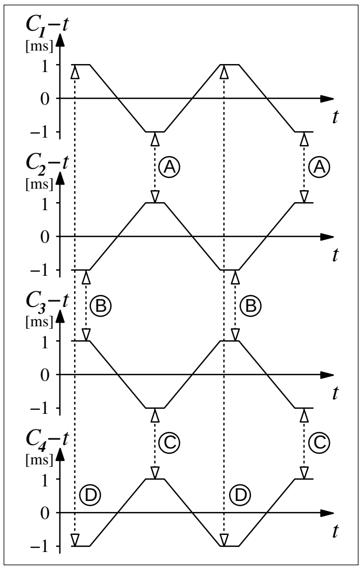


Abb. 7.3: Beispiel ohne globalen Ausgleich

Allgemein kann man sagen, daß die Lösung außerhalb der Grenzen liegen kann (und somit die Uhrenbedingung nicht erfüllt ist), wenn die Uhrengeschwindigkeiten der Prozessoruhren nicht konstant sind. Abb. 7.3 stellt kein realistisches Szenarium dar, aber es zeigt die Problematik exemplarisch auf. Die Diagramme zeigen für jeden Prozessor die Differenz der Prozessoruhr C_i zur physikalischen Zeit t. Die punktierten vertikalen Doppelpfeile Ⓐ bis Ⓓ symbolisieren den Austausch von Nachrichten zwischen den jeweiligen Prozessoren in beiden Richtungen. Es werden die Zeitstempel der Sende- und Empfangsereignisse aufgezeichnet. Die Korrekturverfahren sollen nun die mit den Uhrenfehlern behafteten Zeitstempel so korrigieren, daß die Uhrenbedingung gilt. Die Nachrichtenlaufzeit sei annähernd Null. Hofmanns Verfahren sei ohne Teilintervalle angewandt. Beide Verfahren berechnen bei derart periodischen Szenarien die horizontalen Parallelstreifen zwischen $\overline{O_{ik}} = \max(C_i(e_i^{send}) - C_k(e_k^{recv}))$ und $\underline{O_{ik}} = \min(C_i(e_i^{recv}) - C_k(e_k^{send}))$. Wegen der vernachlässigbaren Nachrichtenlaufzeiten sind beide Werte nahezu identisch, d.h. $\overline{O_{ik}} \simeq \underline{O_{ik}} \simeq C_i(e_k^{send}) + 0 - C_k(e_k^{send})$, d.h. $\overline{O_{ik}}$ und $\underline{O_{ik}}$ berechnen beide, um wieviel die Uhr C_i gegenüber C_k vorgeht. Die Wahl der Zeitpunkte der Nachrichten ergibt nun, daß bei Ⓐ die Uhr C_1 gegenüber C_2 2 ms nachgeht, bei Ⓑ C_2 gegenüber C_3 2 ms, und bei Ⓒ C_3 gegenüber C_4 2 ms, d.h. zusammengenommen geht C_1 gegenüber C_4 6 ms nach. Andererseits ergibt die direkte Ermittlung der Uhrendifferenz zwischen C_1 und C_4 bei Ⓓ, daß C_1 gegenüber C_4 2 ms vorgeht. Dieser Widerspruch bedeutet, daß der Ausgleichsalgorithmus keine Lösung besitzt, da es keine Lösung mit konstanten Uhrengeschwindigkeiten (bei Duda) bzw. konstanten Uhrendifferenzen (bei Hofmann und Hilgers) gibt.

Das von Hofmann und Hilgers in [HH98, Hil96] dargestellte Ausgleichsverfahren benötigt daher neben dem dort angegebenen Kriterium `while (change in any` $\underline{O_{ik}} + \overline{O_{ik}}$`)` das zusätzliche Kriterium `and (all elements` $E_{ik} \geq$ `0)`. Die resultierende Matrix kann nur verwendet werden, wenn alle Elemente der Fehlermatrix nicht negativ bleiben.

Prinzipiell ist Hofmanns Algorithmus zur Berechnung der Uhrendifferenzen Dudas konvexer Hülle vorzuziehen, da die Zeit für die Berechnung der Parallelstreifen proportional der Anzahl der Ereignisse ist, d.h. in der Ordnung $O(\sharp E)$. Auch ist Hofmann und Hilgers Algorithmus zur Berechnung des Ausgleichs wesentlich einfacher, da hierbei nur konstante Uhrenabweichungen und keine linearen Uhrenabweichungen (wie bei Duda und Jézéquel) ausgeglichen werden müssen. Dieser Ausgleich ist dafür für jedes Teilintervall durchzuführen. Da normalerweise die Rechenzeit für den Ausgleichsalgorithmus gegenüber der Rechenzeit für die Berechnung der Uhrendifferenzen vernachlässigbar ist, und da normalerweise die Anzahl der Teilintervalle gering ist, ist es kein Nachteil, daß der Ausgleich mehrfach durchzuführen ist.

Hofmanns Algorithmus benötigt aber zur Beherrschung konstanter, aber unterschiedlicher Uhrengeschwindigkeiten eine Unterteilung des Gesamtintervalls in Teilintervalle und am Ende eine Regressionsanalyse zur Berechnung der Uhrenkorrektur und Gangabweichung über das Gesamtintervall.
Das Beispiel in Abb. 7.4 zeigt, daß diese Regressionsanalyse im Extremfall zu falschen Ergebnissen kommen kann. Daher ist auch hier, wie schon oben erwähnt, eine nachträgliche Kontrolle des Ergebnisses unumgänglich. Rechnet man das Beispiel mit Hofmanns Algorithmus für das Gesamtintervall, dann erhält man für den ersten Prozeß eine Korrektur von 0 ms und für den zweiten Prozeß -0.5 ms, wobei die Laufzeit nicht kleiner als 1.5 ms sein darf, damit die Werte in der Fehlermatrix größer oder gleich Null bleiben. Teilt man dagegen das Intervall in drei gleiche Teile, dann erhält man für den ersten Prozeß die drei Korrekturwerte (0 ms, 0 ms, 0 ms) und für den zweiten Prozeß (1 ms, -2 ms, 1 ms). Die lineare Regression ergibt für beide Prozesse eine konstante Uhrenkorrektur, und zwar 0 ms für beide Prozesse. Für Laufzeitwerte innerhalb von [1.5, 2) werden Nachrichten von Prozeß 2 nach Prozeß 1 in der Mitte des Intervalls kausal falsch dargestellt.

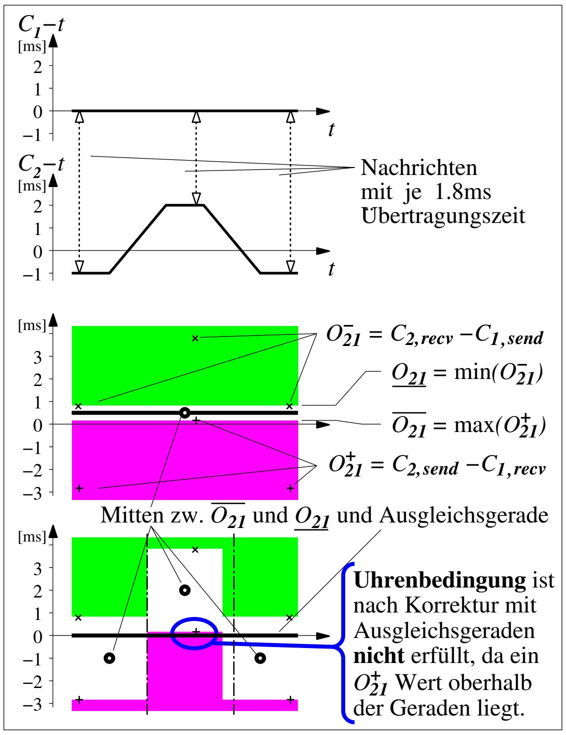


Abb. 7.4: Uhrenkorrektur bei beliebiger Gangabweichung

Beide Beispiele stellen prinzipielle Probleme dar, die auftreten können, sobald die Uhrengeschwindigkeiten der Prozessoruhren nicht hinreichend konstant sind. In der Praxis können Dudas und Hofmanns Algorithmen daher nicht eingesetzt werden, wenn sich die Uhrengeschwindigkeiten der Prozessoruhren im Meßzeitraum hinreichend stark ändern und dadurch bei mindestens einem Prozessorpaar die weiße Fläche in Abb. 3.2 und Abb. 3.3 auf S. 18f so krümmt, daß keine Gerade mehr hineingelegt werden kann. Dies kann z.B. schon bei Zeiträumen im

Sekundenbereich gegeben sein, wenn die Zeitmessung eine XNTP-Synchronisation beinhaltet.

Im folgenden Kapitel werde ich daher ein Verfahren vorstellen, das auf Hofmann und Hilgers Algorithmus aufbaut, aber weder konstanter Uhrengeschwindigkeiten bedarf, noch das soeben dargestellte Problem mit der Regressionsanalyse besitzt.

7.7 Neues Verfahren zur nachträglichen Synchronisation für variable Uhrengeschwindigkeiten

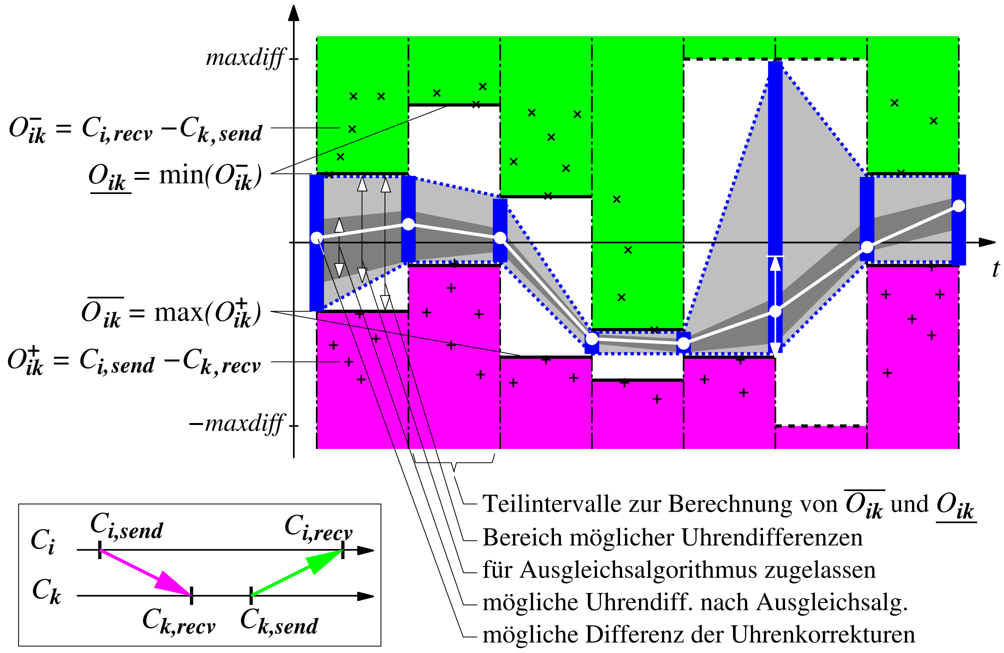


Abb. 7.5: Uhrenkorrektur für variable Uhrengeschwindigkeiten

In Abb. 7.5 auf S. 86 stelle ich ein neues Verfahren zur nachträglichen Synchronisation vor, das die im vorherigen Kapitel geschilderten Grenzen des Verfahrens von Hofmann und Hilgers teilweise überwindet. Im folgenden wird zuerst wie bei Hofmann eine Intervallaufteilung mit anschließender Anwendung des Min/Max-Algorithmus durchgeführt. Anschließend stelle ich die gegenüber Hofmann und Hilgers neue Ausgleichsvorbereitung vor und gebe dann einen Überblick über den restlichen Algorithmus. Nach einer Beschreibung der hierzu auch im Ausgleichsalgorithmus angebrachten Änderungen wird das vollständige Programmschema dieses neuen Verfahrens angegeben. Das Kapitel wird mit einer kurzen Bewertung abgeschlossen.

Intervallaufteilung und Min/Max-Algorithmus

Zuerst wird, wie bei Hofmann, das Gesamtintervall in gleichlange Abschnitte geteilt. Diese Aufteilung geschieht lokal für jeden Prozeß auf der Basis der Prozessoruhren. Anschließend wird für jedes Teilintervall und jedes Prozeßpaar (P_i, P_k) Hofmanns Min/Max-Algorithmus angewandt:[3]

$$O_{ik}^{+} = C_i(e_i^{send}) - C_k(e_k^{recv}) \tag{7.6}$$
$$\overline{O_{ik}} = \max_{(e_i^{send}), e_k^{recv}) \in M} (O_{ik}^{+}) \tag{7.7}$$
$$O_{ik}^{-} = C_i(e_i^{recv}) - C_k(e_k^{send}) \tag{7.8}$$
$$\underline{O_{ik}} = \min_{(e_k^{send}), e_i^{recv}) \in M} (O_{ik}^{-}) \tag{7.9}$$

Die neue Ausgleichsvorbereitung auf den Intervallgrenzen

Im Gegensatz zum Algorithmus von Hofmann und Hilgers wird nun der Ausgleichsalgorithmus nicht auf den Intervallmitten der Auswerteintervalle, sondern nun auf den Intervallgrenzen durchgeführt. Hierzu bedarf es einer zusätzlichen Ausgleichsvorbereitung. Die Differenz $C_i - C_k$, d.h. wieviel C_i gegenüber C_k vorgeht, liegt zwischen $\overline{O_{ik}}$ und $\underline{O_{ik}}$. Dies ist der weiße Bereich zwischen den schwarzen horizontal durchgezogenen Linien. Wenn in einer oder beiden Richtungen keine Nachrichten ausgetauscht wurden, dann wird ein ausreichend großer Wert *maxdiff*, bzw. *−maxdiff* als Grenze angenommen wird. Der Wert von *maxdiff* muß größer sein als jede mögliche Uhrendifferenz. Dies ist in der Abbildung durch die horizontalen gestrichelten Linien symbolisiert. Nun werden die Teilintervalle bei jedem Prozeßpaar aneinandergesetzt und eine gemeinsame Ober- und -Untergrenze für jeweils zwei benachbarte Intervalle gebildet. Hierzu wird auf den Intervallgrenzen jeweils das Maximum der zwei angrenzenden $\overline{O_{ik}}$-Werte, bzw. das Minimum der $\underline{O_{ik}}$-Werte gebildet. Dies ist in der Abbildung durch die breiten vertikalen Balken dargestellt. Das hellgraue Gebiet, das durch diese Balken aufgespannt wird, hat einerseits stückweise lineare Grenzen (punktierte Linien) und garantiert andererseits, daß nach einer Korrektur der Uhrendifferenz mit Werten aus diesem Gebiet die Uhrenbedingung erfüllt ist. Bis jetzt behandelt der Algorithmus nur Prozessorpaare.

Überblick über den restlichen Algorithmus

Der restliche Teil des Algorithmus berechnet nun eine stückweise lineare Korrektur für jede Prozessoruhr, die garantiert, daß die daraus folgende Korrektur der Uhrendifferenzen bei jedem Prozessorpaar innerhalb der hellgrauen Fläche liegt. Hierzu wird an den Intervallgrenzen der Ausgleichsalgorithmus von Hofmann und Hilgers angewandt (im Gegensatz zur Originalarbeit, bei der die Intervallmitten benutzt wurden). Der Algorithmus liefert neue Werte für $\overline{O_{ik}}$ und $\underline{O_{ik}}$. Dies ergibt einen Bereich der garantiert innerhalb des hellgrauen Gebiets liegt. In der Abbildung ist eine mögliche Lösung dunkelgrau dargestellt. Außerdem liefert der Ausgleichsalgorithmus auf jeder Intervallgrenze Uhrenkorrekturwerte für jeden Prozeß. Es wird hierbei garantiert, daß die Differenzen der Korrekturwerte in dem dunkelgrauen Bereich liegen. Eine mögliche Lösung für die resultierende Uhrendifferenzkorrektur ist an den Intervallgrenzen mit weissen Punkten dargestellt. Die weiße Linie ist dann die stückweise lineare Lösung.

Erweiterter und geänderter Ausgleichsalgorithmus

Der Ausgleichsalgorithmus von Hofmann und Hilgers wird hierzu folgendermaßen erweitert, bzw. verbessert:

[3] Die Notation mit den hochgestellten Symbolen + und − wurde wie bei Hofmann angewandt, obwohl hierdurch der kleinere Wert $O_{ik}^{+} = C_i(e_i^{send}) - C_k(e_k^{recv}) = (C_i(e_i^{send}) - C_k(e_i^{send})) - (C_k(e_k^{recv}) - C_k(e_i^{send})) =$ *Uhrendifferenz* **minus** *Nachrichtenlaufzeit* mit einem + gekennzeichnet wird, während der größere Wert $O_{ik}^{-} =$ *Uhrendifferenz* **plus** *Nachrichtenlaufzeit* mit einem − gekennzeichnet wird.

(a) Die in [HH98] angegebene Behandlung fehlender Kommunikationsbeziehungen kann entfallen, wenn man hierzu die Matrix $\overline{O_{ik}}$ mit $+maxdiff$, bzw. $\underline{O_{ik}}$ mit $-maxdiff$ vorbesetzt.

(b) Außerdem wird in [HH98] der Fall offengelassen, daß (innerhalb eines Intervalls) die transitive Hülle der Kommunikationsbeziehungen kein vollständiger Graph ist. In diesem Fall bleibt in der Matrix auch nach Anwendung des Ausgleichsalgorithmus mindestens ein Wert unbestimmt, d.h. im Bereich von $\pm maxdiff$. Hierbei gibt es zwei Möglichketen:

(b1) Der Graph der Kommunikationsbeziehungen ist zusammenhängend. Dann liegt die Ursache, daß die transitive Hülle nicht vollständig ist, darin, daß in einer Richtung zwischen zwei Prozessen Nachrichten fehlen, und daß zwischen diesen Prozessen in dieser Richtung auch keine Nachrichtenkette existiert. Das Problem kann dadurch reduziert werden, daß man bei allen Kommunikationsbeziehungen, bei denen Nachrichten nur in einer Richtung existieren, die fehlende Grenze durch die vorhandene plus oder minus einem Vielfachen des Minimums von $\underline{O_{ik}} - \overline{O_{ik}}$ ersetzt, wobei das Minimum hierbei über alle Prozeßpaare i, k mit Nachrichten in beiden Richtungen gebildet wird. Dies ist durch den weißen Doppelpfeil auf der vorletzten Intervallgrenze in der Abbildung 7.5 dargestellt. Dies heißt, daß hier die obere Spitze des hellgrauen Gebiets auf die Spitze des weißen Pfeils abgesenkt wird. Als Vielfaches sei der Faktor 2 vorgeschlagen. Der Faktor 2 berücksichtigt, daß aufgrund der Gangabweichung der Prozessoruhren innerhalb eines Ausgleichsintervalls der Wert $\min_{i,k}(\underline{O_{ik}} - \overline{O_{ik}})$ geringer sein kann, als die minimale Roundtriptime und, daß beim Setzen der fehlenden Grenze der Abstand zur existierenden Grenze die minimale Roundtriptime nicht unterschreiten sollte, damit die in Abb. 7.3 aufgezeigte Problematik nicht auftritt. Der in (b1) beschriebene Algorithmus wird auch bei nicht zusammenhängenden Graphen durchgeführt, doch ist er dort nicht ausreichend:

(b2) Wenn der Graph der Kommunikationsbeziehungen nicht zusammenhängend ist, oder wenn das Verfahren in (b1) nicht ausreicht, um nach dem Ausgleichsalgorithmus für alle Prozessorpaare eine reale Beschränkung für $\overline{O_{ik}}$ und $\underline{O_{ik}}$ zu besitzen, dann kann dies daran erkannt werden, daß nach Anwendung des Ausgleichsalgorithmus in den Matrizen $\overline{O_{ik}}$ und $\underline{O_{ik}}$ Werte im Bereich von $maxdiff$ (d.h. Betrag $> 0.5\, maxdiff$) stehen bleiben. Dann müssen die hierzu gehörenden Teilintervalle auf der Basis benachbarter Intervalle interpoliert werden. Hierdurch werden dann die $+/-maxdiff$-Werte auf den zwischenliegenden Intervallgrenzen ersetzt. Anschließend ist der Ausgleichsalgorithmus nochmals zu starten. Besteht das Problem beim ersten oder letzten Intervall, dann kann eine horizontale Extrapolation durchgeführt werden.

(c) Da grundsätzlich $\overline{O_{ik}} = -\underline{O_{ki}}$ gilt, und diese Gleichheit durch den Algorithmus auch nicht verletzt wird, kann bei der Implementierung des Ausgleichsalgorithmus die Berechnung von $\underline{O_{ki}}$ entfallen. Sie ist hier nur dargestellt, da sie das Verständnis wesentlich erleichtert.

(d) Der in Kap. 7.6 angegebene fehlende Test auf $(\texttt{E} \geq 0)$, d.h. auf $(\underline{O_{ik}} \geq \overline{O_{ik}})$, wird hinzugefügt, damit sichergestellt wird, daß der Algorithmus abbricht, wenn kein Ausgleich möglich ist.

Vollständiges Programmschema

Ohne Berücksichtigung von (b2) und (c) ergibt dies folgendes Programmschema für jedes Teilintervall:

`for` $i = 1$ `to` n
  `for` $k = 1$ `to` n
    $\underline{O_{ik}} = +maxdiff$
    $\overline{O_{ik}} = -maxdiff$

for ever message (e_k^{send}, e_i^{recv}) do
  $\underline{O_{ik}} = \min(\underline{O_{ik}}, C_i(e_i^{recv}) - C_k(e_k^{send}))$
for ever message (e_i^{send}, e_k^{recv}) do
  $\overline{O_{ik}} = \max(\overline{O_{ik}}, C_i(e_i^{send}) - C_k(e_k^{recv}))$

$mindiff = maxdiff$
for $i = 1$ to n
  for $k = 1$ to n
    if $\underline{O_{ik}} \neq maxdiff$ and $\overline{O_{ik}} \neq -maxdiff$ then $mindiff = \min(mindiff, \underline{O_{ik}} - \overline{O_{ik}})$

for $i = 1$ to n
  for $k = 1$ to n
    if $\underline{O_{ik}} = maxdiff$ and $\overline{O_{ik}} \neq -maxdiff$ then $\underline{O_{ik}} = \overline{O_{ik}} + factor * mindiff$
    if $\overline{O_{ik}} = -maxdiff$ and $\underline{O_{ik}} \neq maxdiff$ then $\overline{O_{ik}} = \underline{O_{ik}} - factor * mindiff$

Mit *factor* sollte das Verhältnis von längster zu kürzester Nachrichtenlaufzeit abgeschätzt werden. Die nun berechneten $\underline{O_{ik}}$ und $\overline{O_{ik}}$ gelten für das Innere der Teilintervalle. Nun werden an den Grenzen der Intervalle die Maxima und Minima der angrenzenden Intervalle gebildet. $\underline{O_{ik}}$ und $\overline{O_{ik}}$ bezeichnen nun diese neuen Werte auf den Intervallgrenzen. Der Ausgleichsalgorithmus ist folgender:

do
  for $i = 1$ to n
    for $k = 1$ to n
      for $z = 1$ to n
        $\underline{O_{ik}} = \min(\underline{O_{ik}}, \underline{O_{iz}} + \underline{O_{zk}})$
        $\overline{O_{ik}} = \max(\overline{O_{ik}}, \overline{O_{iz}} + \overline{O_{zk}})$
      if $\underline{O_{ik}} < \overline{O_{ik}}$ then
        Ausgleich nicht möglich; exit
while (change in any $\underline{O_{ik}}$, $\overline{O_{ik}}$)

Hierbei ist z der Index eines beliebigen Zwischenknotens. Die zentralen Minimum- und Maximumbildungen dieses Algorithmus verschieben $\underline{O_{ik}}$ nach unten, bzw. $\overline{O_{ik}}$ nach oben, indem die Fortpflanzung der Uhrendifferenzen via Zwischenknoten berücksichtigt wird. Da dieser Algorithmus so lange wiederholt wird, bis keine Änderung mehr eintritt, ist somit auch jede Kombination von Zwischenknoten berücksichtigt. Dies heißt, daß nun die Uhrenbedingung für jedes Paar (e_i^j, e_k^l) mit $e_k^l \rightarrow e_i^j$ erfüllt ist, wenn man die Uhren mit Werten zwischen $\overline{O_{ik}}$ und $\underline{O_{ik}}$ einer beliebigen Zeile i korrigiert.

Nach diesem Ausgleichsalgorithmus enthalten die Variablen $\underline{O_{ik}}$ und $\overline{O_{ik}}$ die obere und untere Grenze der dunkelgrauen Bereiche in Abb. 7.5, jeweils auf den Intervallgrenzen. Wenn nicht, dann ist der in (b2) beschriebene Algorithmus anzuwenden. Zur Berechnung der daraus resultierenden Uhrenkorrektur wird das Verfahren von Hofmann und Hilgers ungeändert übernommen:

for $i = 1$ to n
  for $k = 1$ to n
    $O_{ik} = (\underline{O_{ik}} + \overline{O_{ik}})/2$
    $E_{ik} = (\underline{O_{ik}} - \overline{O_{ik}})/2$

Nun wird der Index i_{ref} mit der kleinsten Summe $\Sigma_{k=1}^{n} E_{i_{ref}k}$ bestimmt. Die Korrekturwerte $Corr_k$ für die einzelnen Uhren C_k ergeben sich dann aus der i_{ref}-ten Zeile von O_{ik}:

`for` $k = 1$ `to` n: $\quad Corr_k = O_{i_{ref}k}$

Die in Abb. 7.5 angegebene Uhrendifferenzkorrektur (weißer Punkt) ist dann $-(Corr_i - Corr_k)$.

Bewertung

Folgende Nachteile des Verfahrens von Duda konnten auch durch die in diesem Kapitel dargestellten Verbesserungen nicht beseitigt werden:

- Es benötigt zwei Durchgänge und kann daher zur Online-Visualisierung nicht eingesetzt werden.
- Es ist nicht garantiert, daß das Verfahren eine Lösung liefert. Sowohl bei der Bildung von $\overline{O_{ik}}$ und $\underline{O_{ik}}$ als auch beim Ausgleichsalgorithmus kann festgestellt werden, daß keine Lösung ermittelbar ist, da die Gangabweichung oder die Änderungen der Gangabweichung innerhalb eines Intervalls zu groß waren. Unter Umständen muß das Verfahren mit einer vergrößerten Anzahl von Intervallen nochmals durchgerechnet werden. Dies aber erhöht das Risiko, daß aufgrund fehlender Kommunikationsbeziehungen der Lösungsbereich mit den oben dargestellten Methoden (siehe (b1) und (b2)) eingeschränkt wird und damit u.U. wieder keine Korrektur möglich ist, die die Uhrenbedingung erfüllt.
- Bei großen Datensätzen kann eine Zwischenspeicherung von zwei Intervallen die Kapazität des Memories übersteigen.Es kann somit notwendig werden, daß die Daten zuerst für die Korrekturermittlung und anschließend ein zweites mal für die Durchführung der Korrektur von einem sekundären Datenspeicher (z.B. Festplatte) gelesen werden müssen.
- Uhren, die nur eine geringe Abweichung voneinander haben, können durch das Verfahren stark *verstellt* werden, wenn die Anwendung nur Nachrichten mit großen Nachrichtenlaufzeiten austauscht, insbesondere im unsymmetrischen Fall, d.h. wenn die Nachrichtenlänge und damit die Übertragungsdauer in den beiden Richtungen zwischen zwei Prozessen unterschiedlich ist. Dies ist vor allem bei Client-Server-Programmen gegeben.

7.8 Vergleich mit der geregelten logischen Uhr

Tabelle 7.1 auf Seite 91 vergleicht die verschiedenen Algorithmen. Hierbei wurden die in Kapitel 3.2 auf Seite 16 definierten Anforderungen präzisiert. In Klammern sind zusätzlich die Nummern der zugehörigen Anforderungen aus Kap. 3.2 angegeben. Das Symbol + bedeutet, daß der Algorithmus für den Anwendungsfall uneingeschränkt einsetzbar ist, während das Symbol – zeigt, daß der Algorithmus in diesem Anwendungsfall nicht eingesetzt werden kann. Die Ziffern 1) bis 7) sind Bemerkungen zur Tabelle und geben Einschränkungen bei der Anwendbarkeit der Algorithmen an und sind nachfolgend erläutert. Die Bemerkung 8) analysiert die Faktoren, die bei der Abschätzung der Berechnungsdauer jeweils maßgebend sind.

Anforderung	Duda [DHHB87] 1)	Jézéquel [Jez89a]	Hofmann & Hilgers [HH98]	[HH98] erweitert Kap. 7.7	geregelte logische Uhr
Minimale Fehler bei Zeitdifferenzmessungen zwischen Ereignissen innerhalb eines Prozesses	+	+	+	+	+
Garantiert die Uhrenbedingung für 2 Prozesse bei konstanter Uhrengeschwindigkeit	+	+	3) 4)	3)	+
und bei variabler Uhrengeschwindigkeit	2)	2)	–	3)	+
für mehr als 2 Prozesse bei Uhrengeschwindigkeit =1	7)	+	+	+	+
bei konstanter Uhrengeschwindigkeit (A2)	7)	+	3) 4)	3) 5)	+
bei variabler Uhrengeschwindigkeit (A3)	–	–	–	3) 5)	+
bei Uhrenfehlern in Form von plötzlichen Sprüngen oder bei einer Uhrenauflösung im Bereich der Nachrichtenlaufzeiten oder größer (A3)	–	–	–	–	+
Geringe absolute Synchronisationsfehler (A4)	+	+	+	+	+
Keine Störung des Programmablaufs durch die globale Uhr (A5 und A6)	+	+	+	+	+
Berechnung in einem Durchlauf als Filter möglich	–	–	–	6)	+
Real-Zeit Implementierung mit Verzögerung kleiner als 0.2 sec möglich (A7)	–	–	–	–	+
Komplexität des Algorithmus Berechnungsdauer proportional zu (n=Anzahl der Prozesse, e=Anzahl der Ereignisse, m=Anzahl der Nachrichten, v=Anzahl der Teilintervalle)	mittel $O(m^2)$ $+O(n^3)$ $+O(e)$	mittel $O(m^2)$ $+O(n^3)$ $+O(e)$	gering $O(m)$ $+O(vn^3)$ $+O(e)$ 8)	gering $O(m)$ $+O(vn^3)$ $+O(e)$ 8)	groß $O(m)$ $+O(e \log_2 n)$ $+O(e)$ 8)

Tabelle 7.1: Vergleich der nachträglichen Synchronisationsalgorithmen

1) Bei Dudas Algorithmus muß die Methode der konvexen Hülle eingesetzt werden, damit die Uhrenbedingung garantiert ist.

2) Dudas Algorithmus kann eingesetzt werden, wenn die Mitte der oberen und unteren Hüllkurve aus [DHHB87] zur Uhrenkorrektur benutzt wird. Diese Methode ist weder bei Duda noch bei Jézéquel aufgezeigt, da sie nicht auf beliebig viele Prozessoren erweitert werden kann.

3) Alle auf Hofmanns Algorithmus aufbauende Methoden können nur eingesetzt werden, wenn die Gangabweichungen so gering sind, daß die Teilintervalle so groß gewählt werden können, daß innerhalb der Teilintervalle die transitive Hülle des gerichteten Graphen der Kommunikationsbeziehungen vollständig ist. Wegen der Gangabweichungen ρ_i müssen die Ausgleichsintervalle einiges kürzer als $\mu/2\max_i|\rho_i|$ sein, da sonst an den Intervallgrenzen kein gemeinsamer vertikaler Bereich mehr existiert, durch den die Ausgleichsgerade oder die Ausgleichskurve geführt werden kann. Andererseits sollte die transitive Hülle der Kommunikationsbeziehungen innerhalb jedes Ausgleichsintervalls möglichst vollständig sein, da die Algorithmen in den im Kapitel 7.7 angegebenen Fällen (b), (b1) und vor allem (b2) immer das Risiko enthalten, daß der Algorithmus aufgrund unzulässiger Interpolationen versagt. Bei Hofmann und Hilgers Algorithmus wird kein Ausgleich unter Hinzuziehung der Nachbarintervalle durchgeführt. Daher muß hier die transitive Hülle vollständig sein. Dies bedeutet, daß der Algorithmus bei länger laufenden Anwendungen mit Abschnitten geringer Kommunikation leicht versagen kann. Bei Systemen mit $\rho_{\max} = \max_i|\rho_i| = 10^{-5}$ und mit einer minimalen Nachrichtenlaufzeit $\mu = 200\mu$s ergibt dies, daß in jeder Periode kleiner als 10 sec die transitive Hülle der Kommunikationsbeziehungen möglichst vollständig sein sollte. In derzeitigen schnellen PC-Netzwerken mit einer minimalen MPI-Nachrichtenlaufzeit $\mu = 10\mu$s und gleichem $\rho_{\max}$ reduziert sich diese Periode auf 0.5 sec. Oder beim Einsatz von Low-Level-Kommunikationsmethoden erreicht man derzeit problemlos $\mu = 2\mu$s. Selbst bei einer optimistischen Annahme von $\rho_{\max} = 10^{-6}$ erhält man nur eine Periode im Bereich von 1 sec. Anwendungen, die zwischendurch kommunikationsfreie Zeiten besitzen, die länger als diese Perioden sind, sind somit für diese nachträglichen Synchronisationsalgorithmen nur bedingt geeignet.

4) Die Bildung der Ausgleichsgeraden kann in Extremfällen zu einer Verletzung der Uhrenbedingung führen, vgl. Abb. 7.4 auf S. 85, da die Wahl der Stützstellen der Ausgleichsgeraden nicht garantiert, daß die Ausgleichsgerade nur in *erlaubtem Terrain* verläuft.

5) Die Bemerkung 3) wird insofern abgeschwächt, daß fehlende Kommunikationsbeziehungen über die benachbarten Teilintervalle interpoliert werden.

6) Bei hinreichenden Kommunikationsbeziehungen in jedem Teilintervall genügt es, die Daten für ein Teilintervall zwischenzuspeichern.

7) Duda et al. beschreiben nur zwei Fälle, eine Ring-Topologie und die Topologie, daß jeder Prozeß mit jedem in beiden Richtungen kommuniziert. Für den letzteren Fall gibt er einen Algorithmus an, der analog zu dem von Hofmann und Hilgers ist, nur daß die minimalen Wege für lineare (über die Zeit) statt für konstante Uhrendifferenzen gebildet werden müssen. Dudas Algorithmus kann auch für beliebige Kommunikationsbeziehungen eingesetzt werden. Dies folgt aus der Argumentation bei Hofmann und Hilgers

bzgl. der Einsetzbarkeit ihres Ausgleichsalgorithmus unter der Berücksichtigung, daß eine explizite Behandlung der fehlenden Kommunikationsbeziehungen nicht nötig ist, wie im vorherigen Kapitel dargelegt. Die einzige Voraussetzung ist, daß die transitive Hülle der Relation *Prozeß A schickt Nachricht zu Prozeß B* vollständig ist.

8) Während die Algorithmen von Duda und Jézéquel wegen ihres Anteils $O(m^2)$ in der Regel wesentlich langsamer sind als die neueren Algorithmen, muß man für einen Vergleich der auf Hofmanns Idee beruhenden Algorithmen mit der geregelten logischen Uhr die Proportionalitätskonstanten berücksichtigen. Während bei Hofmann das zweimalige Einlesen[4] aller Ereignisse bei großen Datenmengen, die nicht im Cache gehalten werden können, ein zweites Lesen vom Datenträger erfordert, aber dafür der Algorithmus eine vergleichsweise einfache Programmlogik besitzt, muß bei der geregelten logischen Uhr nur einmal vom Datenträger gelesen werden, aber dafür eine aufwendige Programmlogik durchgerechnet werden. Für $n \leq 512$ sollten die Terme $O(vn^3)$ und $O(e \log_2 n)$ vernachlässigbar sein, da bei Hofmann normalerweise $v \ll e$ ist, und da bei der geregelten logischen Uhr der logarithmische Term nur durch den bezüglich der Gesamtlogik geringen Teil des Sortierens beim Zusammenführen der schon sortierten einzelnen Ereignisströme bei der Ausgabe entsteht. Dadurch kann man diese Terme für $n \leq 512$ als Bestandteil des bei beiden Verfahren vorhandenen Terms $O(e)$ betrachten.

Ein weiterer Unterschied liegt in der Wahl der Referenzuhr. Während die geregelte logische Uhr versucht, alle Uhren an die am weitesten vorgehende anzugleichen, wird bei Hofmann als Referenzuhr diejenige gewählt, bei der zu allen anderen die Uhrendifferenz am schärfsten bestimmt werden kann.

Ein Vorteil von Dudas und Hofmann & Hilgers Ansatz ist, daß er bezüglich der Meßfehler von Nachrichtenlaufzeiten eine Abschätzung berechnet. Bezüglich der prozeßlokalen Fehler ist dagegen die geregelte logische Uhr von Vorteil. Während die geregelte logische Uhr die Fehler für prozeßlokale Zeitintervalle im Bereich der Ganggenauigkeit der Prozessoruhren beläßt, kann bei Dudas und Hofmanns Ansatz der Fehler um Größenordnungen größer sein, wenn die Nachrichten in den beiden Richtungen zwischen jeweils zwei Prozessen sehr unterschiedliche Laufzeiten besitzen. Bei Laufzeitunterschieden von, z.B. 30 ms und einer Gesamtdauer des Traces von 5 s können durch das Verfahren zusätzliche Gangabweichungen im Bereich von 0.6 % erzeugt werden, während die Gangabweichungen der Uhrenquarze im Bereich von 10^{-5} liegt.

[4] Nach dem ersten Einlesen des Tracefiles können die Uhrendifferenzen in den Ausgleichsintervallen und die Ausgleichsgerade über die gesamte Zeit bestimmt werden. Das zweite Einlesen der Daten ist notwendig, um nun die Zeitstempel gemäß der im ersten Durchgang ermittelten Ausgleichsgeraden zu korrigieren.

Kapitel 8

Zusammenfassung

Die geregelte logische Uhr ist eine Methode zum Korrigieren der Zeitstempel mit folgenden Eigenschaften:

- Die korrigierten Zeitstempel erfüllen die Uhrenbedingung.

  Dies heißt, daß ein kausal nachfolgendes Ereignis grundsätzlich einen späteren Zeitstempel als das vorhergehende Ereignis besitzt. Die neuen Zeitstempel sind damit zur Darstellung der Ereignisse in einem Monitorwerkzeug geeignet.

- Etablieren einer globalen Zeit.

  Damit kann sie problemlos für durchschnittliche Zeitmessungen zwischen Ereignissen zweier verschiedener Prozesse eingesetzt werden, z.B. zur Messung der Nachrichtenlaufzeit.

- Die rückwärtige Amortisation garantiert geringe Meßfehler für Zeitspannen zwischen Ereignissen im selben Prozeß.

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

Um Fehler beim Start der geregelten logischen Uhr zu minimieren, ist die Synchronisation vor Beginn einer zu analysierenden Anwendung empfohlen. Hierbei können auch ressourcenarme Synchronisationsmethoden, wie z.B. XNTP, eingesetzt werden. Sie sind meistens zu anderen Zwecken in Workstationclustern schon installiert. Die geregelte logische Uhr 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 logische Uhr damit anderen Methoden überlegen, da diese meist eine geringe Varianz der Uhrengeschwindigkeiten voraussetzen.

Ein typisches Einsatzgebiet ist das Monitoring paralleler Anwendungen in Workstation-Clustern. Da es sich um einen Filter-Algorithmus mit zeitlich begrenzter Pufferung handelt, kann die geregelte logische Uhr nicht nur offline zur Trace-Analyse, sondern auch online innerhalb von Debuggingwerkzeugen zur Trace-Visualisierung eingesetzt werden.

Es wurden die seither existierenden Verfahren zur nachträglichen Uhrensynchronisation vorgestellt. Diese Verfahren setzen hinreichend konstante Uhrengeschwindigkeiten der Prozessoruhren oder hinreichend kurze Programmlaufzeiten der mittels eines Monitorwerkzeuges zu analysierenden parallelen Anwendung voraus. Die seither existierenden Algorithmen sind auch nicht als Filter implementierbar und können somit auch nicht in Online-Monitoren eingesetzt werden. Das Verfahren von Hofmann und Hilgers wurde so erweitert, daß es auch Systeme mit veränderlichen Uhrengeschwindigkeiten in der Regel beherrscht. Anschließend wurden die existierenden Verfahren, das neue erweiterte Verfahren für variable Uhrengeschwindigkeiten und die geregelte logische Uhr mit rückwärtiger Amortisation miteinander verglichen und bewertet.

Anhang A

Liste der verwendeten Symbole

Symbol	Bedeutung	Typ	definiert in	auf Seite
n	Anzahl der Prozesse	$\in \mathbb{N}$	Def. 1	12
$i,\ k,\ m$	Prozeß-Index	$\in \mathbb{N}$	Def. 1	12
$j,\ l$	Ereignis-Index	$\in \mathbb{N}_o$	Def. 1	12
e_i^j	j-te Ereignis im Prozeß i	$\in E_i$	Def. 1	12
E_i	Menge der Ereignisse im Prozeß i		Def. 1	12
E	Menge aller Ereignisse		Def. 1	12
M	Menge der Sende-Empfangs-Paare		Def. 1	12
t	physikalische Uhrzeit	$\in \mathbb{R}$	Def. 2	12
$t(e_i^j)$	Uhrzeit des Ereignisses e_i^j	$E \longrightarrow \mathbb{R}$	Def. 2	12
$\mathcal{C}$	eine beliebige Uhr	$E \longrightarrow \mathbb{R}$	Def. 2	12
C_i	Prozessoruhr des Prozesses i als Funktion der Zeit t	$\mathbb{R} \longrightarrow \mathbb{R}$	Def. 2	12
$C(e_i^j)$	globale Prozessoruhrzeit des Ereignisses e_i^j	$E \longrightarrow \mathbb{R}$	Def. 2	12
$\frac{d\mathcal{C}(t)}{dt}$	momentane Uhrengeschwindigkeit	$\mathbb{R} \longrightarrow \mathbb{R}$	Def. 2	12
$\rho(t)$	momentane Gangabweichung	$\mathbb{R} \longrightarrow \mathbb{R}$	Def. 2	12
$\rho_{\max}$	Ganggenauigkeit und relative Gangungenauigkeit	$\in \mathbb{R}$	Def. 2 und Satz 17	12 66
e	Synchronisationsfehler	$\in \mathbb{R}$	Def. 3	13
T	globale Zeit zu der C_i synchronisiert ist	$\mathbb{R} \longrightarrow \mathbb{R}$	Bem. 2	13
e'	interner Synchronisationsfehler	$\in \mathbb{R}$	Def. 5	13
$\tilde{\rightarrow}$	Relation *geschah direkt zuvor* *(happened directly before)*	$\subset E \times E$	Def. 6	14
$\rightarrow$	Relation *geschah zuvor* *(happened before)*	$\subset E \times E$	Def. 6	14
LC_i^{int}	Lamports logische Uhr im Prozeß i	$E_i \longrightarrow \mathbb{N}_o$	Alg. 1	23
LC^{int}	Lamports logische Uhr	$E \longrightarrow \mathbb{N}_o$	Alg. 1	23
$-e_i^-$	eine untere Grenze von $C_i - T$	$\in \mathbb{R}$	Def. 8	25
e_i^+	eine obere Grenze von $C_i - T$	$\in \mathbb{R}$	Def. 8	25

Symbol	Bedeutung	Typ	definiert in	auf Seite
μ	minimale Nachrichtenlaufzeit	$\in \mathbb{R}$	Kap. 4.2	25
LC_i	einfache logische Uhr im Prozeß i	$E_i \longrightarrow \mathbb{R}$	Alg. 2	25
LC	einfache logische Uhr	$E \longrightarrow \mathbb{R}$	Alg. 2	26
δ_i	minimaler Abstand zweier Ereignisse in Prozeß i	$\in \mathbb{R}$	Alg. 2	25
$\mu_{k,i}$	minimale Laufzeit einer Nachricht von Prozeß k nach Prozeß i	$\in \mathbb{R}$	Alg. 2	26
$-e^-$	eine untere Grenze aller $C_i - T$	$\in \mathbb{R}$	Gl. (4.7)	27
e^+	eine obere Grenze aller $C_i - T$	$\in \mathbb{R}$	Gl. (4.7)	27
LC'_i	geregelte logische Uhr im Prozeß i	$E_i \longrightarrow \mathbb{R}$	Alg. 3	29
LC'	geregelte logische Uhr	$E \longrightarrow \mathbb{R}$	Alg. 3	29
γ_i^j	Stellgröße des Regelkreises	$\in [0,1]$	Alg. 3	29
P_i	Prozeß i		Kap. 4.4.2	33
$\gamma_{\max}$	eine obere Schranke für die Stellgröße γ_i^j	$\in \mathbb{R}$	Satz 7	35
$\gamma_{\mathcal{A}}$	Stellgröße des Reglerelements $\mathcal{A}$	$\in \mathbb{R}$	Kap. 4.4.3	35
$\gamma_{\mathcal{B}}$	Stellgröße des Reglerelements $\mathcal{B}$	$E \longrightarrow \mathbb{R}$	Abb. 4.10	37
C_i^*	Hilfsuhr für Reglerelement $\mathcal{C}$	$E \longrightarrow \mathbb{R}$	Kap. 4.4.5	38
$\gamma_{i,\mathcal{C}}$	Stellgröße des Reglerelements $\mathcal{C}$	$E_i \longrightarrow \mathbb{R}$	Abb. 4.11	38
$\gamma_{\mathcal{D}}$	Stellgröße des Reglerelements $\mathcal{D}$	$\in \mathbb{R}$	Kap. 4.4.6	39
LC_i^b	eine Uhr innerhalb des Alg. 5	$E_i \longrightarrow \mathbb{R}$	Alg. 5	40
LC_i^{bs}	eine Uhr innerhalb des Alg. 5	$E_i \longrightarrow \mathbb{R}$	Alg. 5	40
LC_i^A	rückwärtig amortisierte geregelte logische Uhr im im Prozeß i	$E_i \longrightarrow \mathbb{R}$	Alg. 5 und Def. 9	40 43
A_{def}	Amortisationsfaktor	$\in \mathbb{R}_o^+$	Alg. 5	40
A	Amortisationsgradient	$\in \mathbb{R}_o^+$	Alg. 5	41
LC^A	rückwärtig amortisierte geregelte logische Uhr	$E \longrightarrow \mathbb{R}$	Def. 9	43
d_M	Nachrichtenlaufzeit	$E \times E \longrightarrow \mathbb{R}$	Kap. 5.1	44
d_R	Zeitintervall zwischen zwei aufeinanderfolgenden Ereignissen im gleichen Prozeß und das letztere ist ein Empfangsereignis	$E \times E \longrightarrow \mathbb{R}$	Kap. 5.1	44
$d_{S/I}$	Zeitintervall zwischen zwei aufeinanderfolgenden Ereignissen im gleichen Prozeß und das letztere ist ein Sendeereignis oder ein internes Ereignis	$E \times E \longrightarrow \mathbb{R}$	Kap. 5.1	44
d^t	exakt gemessenes Zeitintervall	$E \times E \longrightarrow \mathbb{R}$	Kap. 5.1	44
d^C	Zeitintervall gemessen mittels Prozessoruhren C_i	$E \times E \longrightarrow \mathbb{R}$	Kap. 5.1	44
$d^{LC'}$	Zeitintervall gemessen mittels der geregelten logischen Uhr LC'	$E \times E \longrightarrow \mathbb{R}$	Kap. 5.1	44
d^{LC^A}	Zeitintervall gemessen mittels der rückwärtig amortisierten geregelten logischen Uhr LC^A	$E \times E \longrightarrow \mathbb{R}$	Kap. 5.1	44
$e^{LC'}$	absoluter Fehler von LC'	$E \times E \longrightarrow \mathbb{R}$	Gl. (5.1)	45
$\epsilon^{LC'}$	relativer Fehler von LC'	$E \times E \longrightarrow \mathbb{R}$	Gl. (5.2)	45
e^{LC^A}	absoluter Fehler von LC^A	$E \times E \longrightarrow \mathbb{R}$	wie Gl. (5.1)	45
ϵ^{LC^A}	relativer Fehler von LC^A	$E \times E \longrightarrow \mathbb{R}$	wie Gl. (5.2)	45
$\bar{A}$	maximaler Amortisationsgradient	$\in \mathbb{R}^+$	Gl. (5.12)	50
$\tilde{\kappa_a}$	Koeffizient zur Berechnung von $\bar{A}$	$\in \mathbb{R}^+$	Gl. (5.13)	50
$\tilde{\kappa_b}$	Koeffizient zur Berechnung von $\bar{A}$	$\in \mathbb{R}^+$	Gl. (5.14)	50

Symbol	Bedeutung	Typ	definiert in	auf Seite
$\check{A}$	maximaler Amortisationsgradient ohne Anfangssynchronisation	$\in \mathbb{R}^+$	Gl. (5.16)	50
$\check{\kappa}$	Koeffizient zur Berechnung von $\check{A}$	$\in \mathbb{R}^+$	Gl. (5.17)	50
τ	zeitlicher Abstand zweier Ereignisse in Szenarium 2	$\in \mathbb{R}^+$	Szen. 2	51
D	anfängliche Uhrendifferenz	$\in \mathbb{R}^+$	Szen. 2	51
ρ	Gangabweichung von C_1 gegenüber C_0 in den Szenarien 2 bis 4	$\in \mathbb{R}^+$	Szen. 2	51
O_j	Offset	$\in \mathbb{R}$	Gl. (5.21)	52
Ω_j	Relation Offset zu D	$\in \mathbb{R}$	Gl. (5.22)	52
θ	Relation τ zu D	$\in \mathbb{R}^+$	Gl. (5.23)	52
R	relativierte Uhrenabweichung	$\in (1, \infty)$	Gl. (5.24)	52
Ω_∞	Grenzwert zu Ω_j	$\in \mathbb{R}$	Gl. (5.25)	53
o_j	zum Grenzwert Ω_∞ relativiertes Ω_j	$\in \mathbb{R}$	Gl. (5.26)	53
f	Parameter zur Analyse des Funktionsverlaufs von o_j und v_j	$\in \mathbb{R}^+$	Gl. (5.28)	53
v_j	Inverse zu o_j	$\in \mathbb{R}$	Gl. (5.30)	53
ϵ	maximal tolerierbaren Fehlerrate	$\in \mathbb{R}^+$	vor Gl. (5.11)	49
ε	eine Konstante	$\in (0, 1)$	Beweis zu (5.34)-(5.37)	55
O_∞	Grenzwert zu O_j	$\in \mathbb{R}$	Gl. (5.32, 5.33)	59
τ_j	Abstand zweier aufeinanderfolgender Ereignisse in Szen. 4	$\in \mathbb{R}^+$	Szen. 4	64
$\tau_{\min}$	Minimum aller τ_j	$\in \mathbb{R}^+$	Szen. 4	64
$\tau_{\max}$	Maximum aller τ_j	$\in \mathbb{R}^+$	Szen. 4	64
κ	Koeffizient $\tau_{\max}/\tau_{\min}$	$\in (1, \infty)$	Szen. 4	64
$\rho_{\max}$	obere Grenze für die absolute Gangabweichungen der Prozessoruhren C_i	$\in \mathbb{R}^+$	Satz 17	66
$\bar{A}_i$	maximaler Amortisationsgradient	$\in \mathbb{R}^+$	Satz 17	66
$\bar{\kappa}_{a,i}$	Koeffizient zur Berechnung von $\bar{A}_i$	$\in \mathbb{R}^+$	Satz 17	66
$\bar{\kappa}_{b,i}$	Koeffizient zur Berechnung von $\bar{A}_i$	$\in \mathbb{R}^+$	Satz 17	66
$\check{A}_i$	maximaler Amortisationsgradient ohne Anfangssynchronisation	$\in \mathbb{R}^+$	Satz 17	66
$\check{\kappa}_i$	Koeffizient zur Berechnung von $\check{A}_i$	$\in \mathbb{R}^+$	Satz 17	66
$\Delta E/\Delta t$	Anzahl der Ereignisse pro Zeitintervall	$\in \mathbb{R}^+$	Kap. 6.1	69
`-maxerr`	Option zur Angabe von A_{def} in der Implementierung der geregelten logischen Uhr in %	$\in \mathbb{R}^+$	Abb. 6.3	72
`-mmd`	Option zur Angabe von μ in μs	$\in \mathbb{R}^+$	Abb. 6.3	72
`-cldiff`	Option zur Angabe der erwarteten maximalen Uhrendifferenz in μs	$\in \mathbb{R}^+$	Abb. 6.3	72
`us`	ASCII Notation für $\mu\mathrm{s} = 10^{-6}\mathrm{sec}$		Abb. 6.3	72
$\sharp E$	Anzahl der Ereignisse	$\in \mathbb{N}$		85
$\overline{O_{ik}}$	Maximum der negierten Laufzeiten der Nachrichten von Prozeß i nach Prozeß k	$\in \mathbb{R}$	zu Abb. 7.3 und Gl. (7.7)	84 87
$\underline{O_{ik}}$	Minimum der Laufzeiten der Nachrichten von Prozeß k nach	$\in \mathbb{R}$	zu Abb. 7.3	84

Symbol	Bedeutung	Typ	definiert in	auf Seite
	Prozeß i		und Gl. (7.9)	87
O_{ik}^{+}	negierte Laufzeiten der Nachrichten von Prozeß i nach Prozeß k	$M \longrightarrow \mathbb{R}$	Gl. (7.6)	87
O_{ik}^{-}	Laufzeiten der Nachrichten von Prozeß k nach Prozeß i	$M \longrightarrow \mathbb{R}$	Gl. (7.8)	87
$maxdiff$	ein Wert der garantiert größer ist als alle möglichen Uhrendifferenzen	$\in \mathbb{R}^{+}$	Kap. 7.7	87
$mindiff$	minimale Differenz zwischen $\underline{O_{ik}}$ und $\overline{O_{ik}}$	$\in \mathbb{R}$	Kap. 7.7	89
$factor$	Abschätzung des Verhältnisses von längster zu kürzester Nachrichtenlaufzeit	$\in (1, \infty)$	Kap. 7.7	89
O_{ik}	Abschätzung für die Uhrendifferenz $C_i - C_k$	$\in \mathbb{R}$	Kap. 7.7	89
E_{ik}	Abschätzung des absoluten Fehlers von O_{ik}	$\in \mathbb{R}_{o}^{+}$	Kap. 7.7	89
$Corr_k$	Uhrenkorrektur für C_k	$\in \mathbb{R}$	Kap. 7.7	90
e	Anzahl der Ereignisse	$\in \mathbb{N}$	Kap. 7.8	90
m	Anzahl der Nachrichten	$\in \mathbb{N}$	Kap. 7.8	90
v	Anzahl der Teilintervalle	$\in \mathbb{N}$	Kap. 7.8	90

Literaturverzeichnis

[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.

[Bol79] Béla Bollobás. *Graph Theory*. Springer, New York, 1979.

[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.

[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.

[HH98] Richard Hofmann and Ursula Hilgers. Theory and tool for estimating global time in parallel and distributed systems. In *Proc. 6th Euromicro Workshop on Parallel and Distributed Processing*, pages 173–179, Jan. 1998.

[Hil96] Ursula Hilgers. Theorie und Werkzeug zur Erzeugung globaler Zeitstempel aus lokalen Ereignisspuren. Diplomarbeit, Universität Erlangen-Nürnberg, IMMD VII, Okt. 1996. http://rzsunhome.rrze.uni-erlangen.de:81/~unrz74/inhalt.ps.

[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.

[MPI95] MPI Forum. *MPI: A Message-Passing Interface Standard*, June 12, 1995. http://www.mcs.anl.gov/mpi/.

[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.

[NA95] Wolfgang E. Nagel and Alfred Arnold. Performance visualization of parallel programs: The PARvis environment. Technical report, Forschungszentrum Jülich, 1995.
http://www.kfa-juelich.de/zam/PT/ReDec/SoftTools/PARtools/PARvis.html.

[NAW+96] Wolfgang E. Nagel, Alfred Arnold, Michael Weber, Hans-Christian Hoppe, and Karl Solchenbach. VAMPIR: Visualization and analysis of MPI resources. *Supercomputer 63*, XII(1):69–80, January 1996.
auch als Interner Bericht KFA-ZAM-IB-9528, Forschungszentrum Jülich,
ftp://ftp.zam.kfa-juelich.de/pub/zamdoc/ib/ib-95/ib-9528.ps.

[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.

[Rab96] Rolf Rabenseifner. Die geregelte logical Clock – Definition, Simulation und Anwendung. RUS - 30, Rechenzentrum der Universität Stuttgart, Mai 1996.
http://www.uni-stuttgart.de/People/rabenseifner/log_clock_rus30.html.

[Rab97] Rolf Rabenseifner. The controlled logical clock – a global time for trace based software monitoring of parallel applications in workstation clusters. In *Proceedings of the 5th Euromicro Workshop on Parallel and Distributed Processing – PDP '97*, pages 477–484. IEEE CS, Jan. 1997.
http://www.uni-stuttgart.de/People/rabenseifner/log_clock_pdp97.html.

[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.

[SC90] Frank Schmuck and Flaviu Cristian. Continuous clock amortization need not affect the precision of a clock synchronization algorithm. In *Proceedings of the 9th Annual ACM Symposium on Principles of Distributed Computing, Quebec city, Canada, 22-24 August, 1990*, pages 133–143. ACM, 1990. Also as Technical Report RJ7290, IBM, January 1 1990.

[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 T. Anthony Marsland. 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.

Lebenslauf

3. Juli 1956:	Rolf Rabenseifner, geboren in Stuttgart.
1963 bis 1966:	Friedensschule Stuttgart (Grundschule).
1966 bis 1975:	Besuch des Schickhardt-Gymnasiums in Stuttgart.
13. Mai 1975:	Abitur.
1975 bis 1983:	Studium an der Universität Stuttgart.
1976 bis 1977:	Studienunterbrechung wegen Grundwehrdienst.
31. Okt. 1983:	Erster Abschluß mit Diplom in Mathematik.
15. Nov. 1983:	Zweiter Abschluß mit erstem Staatsexamen in Mathematik und Physik.
Seit 1984:	Wissenschaftlicher Mitarbeiter am Rechenzentrum der Universität Stuttgart / Höchstleistungsrechenzentrum Stuttgart (HLRS).
Seit 28. Mai 1988:	Verheiratet mit Annerose Volz-Rabenseifner, zwei Kinder, Katja (*1992) und Jan (*1994)
Jan.-April 1999:	Forschungsaufenthalt am Zentrum für Hochleistungsrechnen an der Technischen Universität Dresden.

Hompage: `http://www.hlrs.de/people/rabenseifner/`
e-mail: `rabenseifner@rus.uni-stuttgart.de`
`rabenseifner@hlrs.de`