Ausarbeitung

Verteilte Snapshots

im Rahmen des Seminars Parallele und verteilte Programmierung

Matthias Bedarff

Themensteller: Prof. Dr. Herbert Kuchen
Betreuer: Dipl.-Wirt. Inform. Michael Poldner
Institut für Wirtschaftsinformatik
Praktische Informatik in der Wirtschaft

Inhaltsverzeichnis

1 Einleitung 1

2 Grundlagen 2

2.1 Globale Zustände in verteilten Systemen 2

2.2 Modell eines verteilten Systems 4

2.2.1 Verteilte Systeme 4

2.2.2 Konsistente Schnappschüsse 6

3 Festhalten von globalen Zuständen 9

3.1 Chandy-Lamport Algorithmus 9

3.1.1 Terminierung des Chandy-Lamport-Algorithmus 11

3.1.2 Verteilung der aufgezeichneten lokalen Zustände 11

3.1.3 Aufwand des Chandy-Lamport Algorithmus 12

3.2 Eigenschaften festgehaltener Schnappschüsse 12

3.2.1 Lokale Zustände zu einem globalen Schnappschuss zusammensetzen ... 14

3.2.2 Beweis der Behauptung, dass der festgehaltene Schnappschuss einen erreichbaren globalen Zustand darstellt 15

3.2.3 Gültigkeit globaler Prädikate in Schnappschüssen 17

3.3 Abweichende Annahmen bezüglich der Nachrichtenkanäle 18

4 Zusammenfassung 20

Literaturverzeichnis 21

1 Einleitung

„Computersysteme durchleben eine Revolution“ [Ta03, S. 19]. Mit diesem Zitat lässt sich die Entwicklung moderner Computersysteme in den letzten Jahrzehnten beschreiben. Bis in die Mitte der 80er-Jahre waren Computer groß und teuer, sodass Unternehmen – wenn überhaupt – nur wenige Computer besaßen, welche unabhängig voneinander arbeiteten.

Durch die Fortschritte bei der *Entwicklung von Mikroprozessoren* wurden Computer immer günstiger. Dazu kam die *Entwicklung von leistungsfähigen Computernetzwerken*, welche es nun ermöglichte viele, günstige Computer zu Computersystemen zusammenzustellen, die über Hochgeschwindigkeits-Netzwerke miteinander verbunden sind. Diese Art von Computersystemen nennt man *verteilte Systeme*. Verteile Systeme können sich nur über einen kleinen geographischen Bereich erstrecken oder auch die gesamte Erde umspannen. Im ersten Fall sind die einzelnen Computer meist über ein lokales Netzwerk miteinander verbunden, im zweiten Fall über globale Weitverkehrsnetzwerke wie das Internet.

Da ein verteiltes System aus vielen einzelnen Computern besteht, die jeweils einen eigenen lokalen Zustand besitzen, besteht ein großes Interesse daran, zu erfahren, in welchem Zustand sich das ganze verteilte System zu einem gewissen Zeitpunkt gerade befindet. Dieser Zustand wird *globaler Zustand* und ein festgehaltener globaler Zustand *Schnappschuss* genannt [Ta03, S. 295]. In dieser Ausarbeitung wird der Algorithmus von CHANDY und LEMPORT zur Erstellung von globalen Schnappschüssen vorgestellt.

Nach dieser Einleitung werden zunächst im Kapitel 2 die Grundlagen verteilter Systeme vorgestellt. Dies umfasst erstens die Motivation für Schnappschüsse und zweitens die Modellierung von verteilten Systemen, Zustandsübergängen durch Ereignisse und die Definition von konsistenten Schnappschüssen. Im Kapitel 3 wird darauf die Erfassung von globalen Schnappschüssen erläutert. Hierzu wird der Chandy-Lamport Algorithmus vorgestellt und bewiesen, dass der aufgezeichnete Schnappschuss einen konsistenten und erreichbaren globalen Zustand darstellt, dessen globale Prädikate auch im Zustand nach Terminierung des Algorithmus gelten. Außerdem soll darauf eingegangen werden, welche Modifikationen des Algorithmus existieren, wenn nicht alle Anforderungen an die Nachrichtenkanäle erfüllt sind. Schließlich endet diese Ausarbeitung im Kapitel 4 mit einer Zusammenfassung der vorgestellten Modelle und Algorithmen.

2 Grundlagen

In diesem Grundlagenkapitel wird zunächst im ersten Unterkapitel für das Erstellen von globalen Schnappschüssen motiviert. Das zweite Unterkapitel behandelt die Modellierung von verteilten Systemen, ereignisabhängigen Zustandsübergängen und konsistenten Schnappschüssen.

2.1 Globale Zustände in verteilten Systemen

Es existieren eine Reihe von Gründen, warum es sinnvoll ist, globale Zustände verteilter System in Form von Schnappschüssen festzuhalten:

- Ermittlung *globaler Prädikate*: Ein globales Prädikat ist eine Eigenschaft eines verteilten Systems, die sich in Zukunft – auch nach weiteren Berechnungen – nicht mehr ändert. Sind bspw. in einem Schnappschuss die lokalen Berechnungen aller Prozesse unterbrochen und es befinden sich keine Nachrichten in den Nachrichtenkanälen, dann ist entweder ein Deadlock aufgetreten oder die Berechnung wurde erfolgreich beendet [Ta03, S. 296].
- Erstellung von *Rücksetzpunkten*: Vor allem bei lang andauernden Berechnungen besteht ein großes Interesse daran, Sicherungen während einer Berechnung zu erstellen, damit die Berechnung wieder aufgenommen werden kann, falls ein Prozess oder Nachrichtenkanal abstürzt.
- *Testen* von verteilten Anwendungen: Während der Entwicklung und Wartung von verteilten Anwendungen ist es von großer Bedeutung, globale Schnappschüsse zum Debuggen und Testen zur Verfügung zu haben.

Da die Prozesse auf physikalisch getrennter Hardware ablaufen, haben diese keinen Zugriff auf einen *gemeinsamen Speicher*, der ständig einen aktuellen globalen Zustand enthalten würde, oder eine *gemeinsame Uhr*, sodass kein perfekt synchronisierter Schnappschuss erfasst werden kann [KR95, S. 224]. Es ist auch nicht möglich, dass ein Prozess als Zeitgeber für die restlichen Prozesse agiert, weil bei den Übertragungen über die verschiedenen Nachrichtenkanäle zeitliche Abweichungen auftauchen würden. Wird dies nicht beachtet, so können unberechenbare Fehler auftreten. Verdeutlicht wird dies im folgenden Beispiel, das einen Schnappschuss einer Überweisung darstellt:

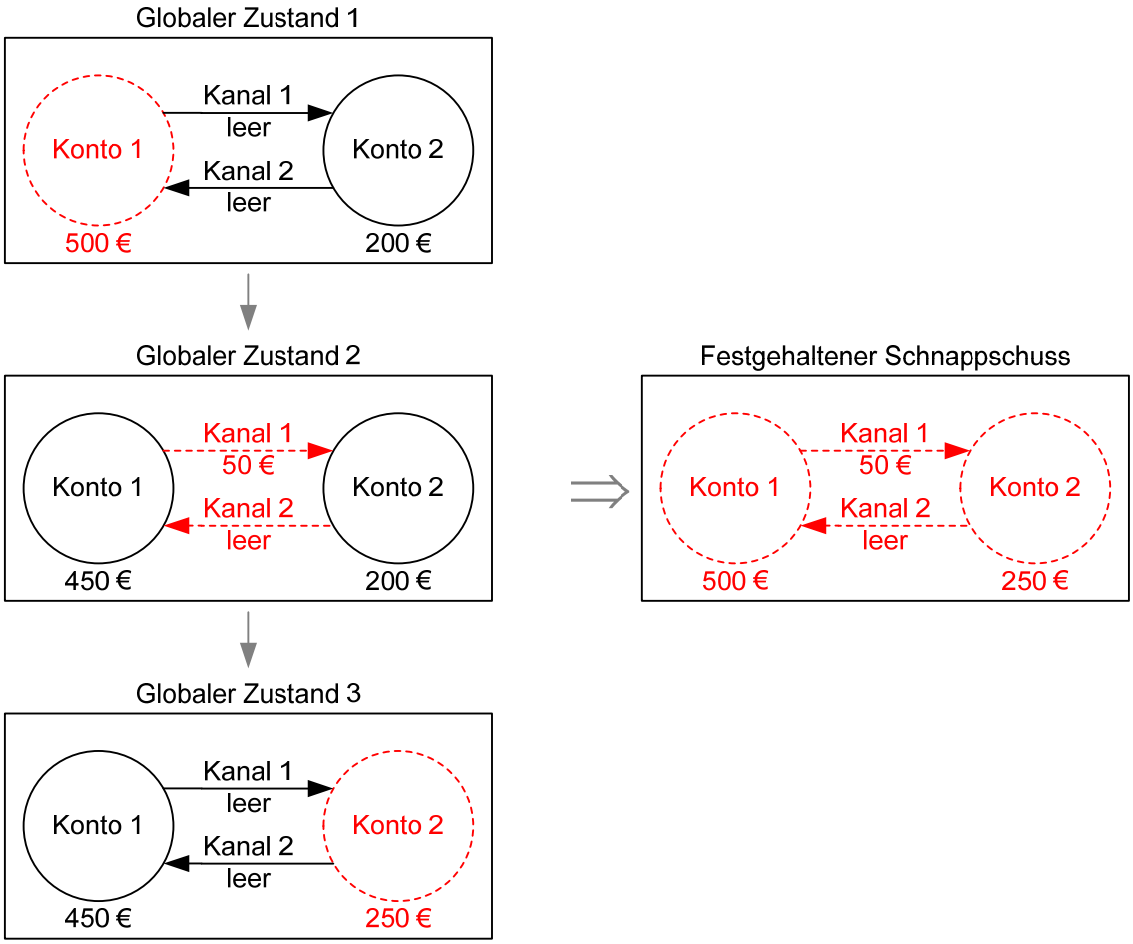

Abb. 1: Schnappschuss einer Überweisung

Das Beispiel zeigt die Erstellung eines Schnappschusses von einer Überweisung von 50 Euro von Konto 1 zu Konto 2. Hierbei steht der Begriff Konto für einen unabhängigen Prozess eines verteilten Systems, welcher die Funktionalitäten eines Kontos anbietet. Da die Erfassungszeitpunkte nicht synchronisiert sind, wurden die einzelnen Teile des verteilten Systems – die rot gestrichelten Bereiche in der Abbildung – zu verschiedenen globalen Zuständen festgehalten. Es fällt auf, dass die Summe der Beträge im festgehaltenen Zustand nicht den Summen in den einzelnen globalen Zuständen entspricht. Das Ausmaß des Problems wird deutlich, wenn das System abstürzt und der festgehaltene Zustand zur Wiederherstellung verwendet wird: Nach Abschluss der Berechnung wäre der Kontostand 1 (wieder) 500 Euro und 300 Euro der Kontostand 2.

Damit soeben beschriebene Probleme nicht auftreten, wird in dieser Ausarbeitung der *Chandy-Lamport Algorithmus* vorgestellt, welcher es ermöglicht, konsistente Schnappschüsse von verteilten Systemen zu erstellen, ohne die Berechnung des verteilten Systems zu beeinflussen.

2.2 Modell eines verteilten Systems

In diesem Unterkapitel wird nun das Modell eines verteilten Systems vorgestellt, das im Folgenden dieser Ausarbeitung verwendet wird. Außerdem wird gezeigt, wie ein verteiltes System durch Ereignisse in einen anderen Zustand übergeht und es werden die Vorraussetzungen für konsistente Schnappschüsse dargestellt.

2.2.1 Verteilte Systeme

Ein verteiltes System besteht aus einer begrenzten Anzahl von Prozessen und einer begrenzten Anzahl von Nachrichtkanälen, die die einzelnen Prozesse miteinander verbinden. Für jeden Kanal gelten die folgenden Annahmen [CL85, S. 65]:

1. Er ist *gerichtet*, d. h. es ist nur eine Übertragung in eine Richtung möglich. Soll ein Nachrichtenaustausch in beide Richtungen möglich sein, so müssen zwei entgegengesetzt gerichtete Kanäle vorhanden sein.
2. Der *Nachrichtenpuffer ist unbegrenzt*. Dies hat zur Folge, dass keine Nachrichten aufgrund eines Pufferüberlaufs verloren gehen können.
3. Es existiert eine *Verzögerung* beim Nachrichtenversand, die für jede Nachricht unterschiedlich lang sein kann, aber im endlichen Bereich liegt.
4. Die *Empfangsreihenfolge* der Nachrichten entspricht der Reihenfolge, in der die Nachrichten abgesendet wurden, sodass ein Kanal eine FiFo (engl. first in, first out) Warteschlange darstellt.

Zu Beginn einer verteilten Berechnung ist jeder Kanal leer. Der Zustand eines einzelnen Kanals ist die Menge der noch nicht empfangenen Nachrichten. Diese Teilmenge ergibt sich aus der Menge aller Nachrichten, die seit Beginn der Berechnung verschickt wurden, abzüglich der Menge an Nachrichten, die bereits empfangen wurden.

Ein Prozess dagegen besitzt einen Anfangszustand, sodass sich der Zustand eines Prozesses aus seinem Anfangszustand und allen danach eingetretenen Ereignissen ergibt. Ein *Ereignis* ist eine unteilbare Aktivität, die zu einem Prozess gehört und höchstens einen Kanal betrifft. Ist nun ein Kanal betroffen, dann liegt der Fall vor, dass ein Prozess eine Nachricht entweder über einen ausgehenden Kanal versendet oder über einen eingehenden Kanal empfängt. Es wird also niemals gleichzeitig empfangen und gesendet. Formal lässt sich ein Ereignis *e* für einen Prozess *p* als Tupel beschreiben:

$$e = \langle p, s, s', M, c \rangle \tag{1}$$

Hierbei ist *s* der lokale Zustand von *p* unmittelbar vor dem Eintreten des Ereignisses *e* und *s'* ist der Zustand unmittelbar nach dem Ereignis. Ist ein Kanal betroffen, dann ist *c* ein durch das Ereignis veränderter Kanal und *M* die bei der Veränderung empfangene bzw. versendete Nachricht. Wenn dagegen kein Kanal betroffen ist, dann besitzen *c* und *M* den Wert *null*.

Ein globaler Zustand setzt sich nun aus den einzelnen lokalen Zuständen aller Prozesse und aller Kanäle zusammen [CL85, S. 66]. Hieraus folgt für den globalen Anfangszustand, dass sich alle Prozesse in ihrem jeweiligen Anfangszustand befinden und alle Kanäle leer sind. Um nun die Auswirkung eines Ereignisses *e* auf einen beliebigen globalen Zustand zu beschreiben, definieren wir eine Funktion *next*(*S*, *e*), die den nächsten globalen Zustand unmittelbar nach dem Ereignis *e* darstellt. *next*(*S*, *e*) ist nur dann definiert, wenn ein Ereignis *e* in einem globalen Zustand *S* eintreten kann, d. h. wenn sich der Prozess *p* gerade in dem Zustand *s* befindet. Es sind zwei Fälle zu unterscheiden:

1. Kanal *c* geht aus dem Prozess *p* heraus: Dann ist der lokale Zustand von Kanal *c* eine Nachrichtenfolge, deren letztes Element die Nachricht *M* sein muss. Der nächste globale Zustand *next*(*S*, *e*) entspricht dem Zustand *S* mit zwei Ausnahmen: *p* befindet sich in dem lokalen Zustand *s'* und der Zustand des Kanals *c* entspricht der Nachrichtenfolge aus *S*, wobei *M* an das Ende der Folge eingefügt wurde.

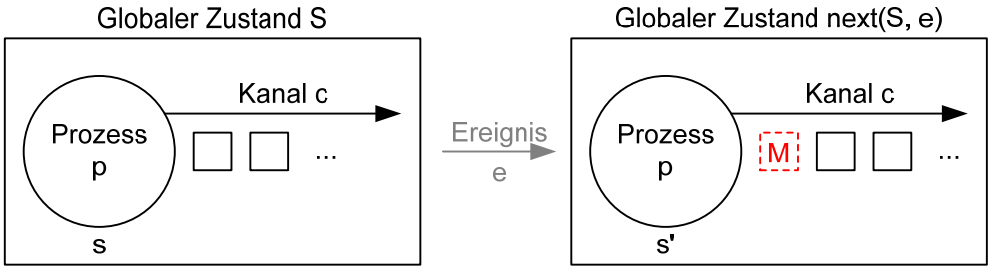

Abb. 2: Auswirkungen eines Ereignisses *e* bei einem ausgehenden Kanal

2. Kanal *c* geht in den Prozess *p* herein: Dann ist der lokale Zustand von Kanal *c* eine Nachrichtenfolge, deren erstes Element die Nachricht *M* sein muss. Der nächste globale Zustand *next*(*S*, *e*) entspricht dem Zustand *S* mit zwei Ausnahmen: *p* befindet sich in dem lokalen Zustand *s'* und der Zustand des Kanals *c* entspricht der Nachrichtenfolge aus *S*, wobei die *M* aus der Folge entfernt wurde.

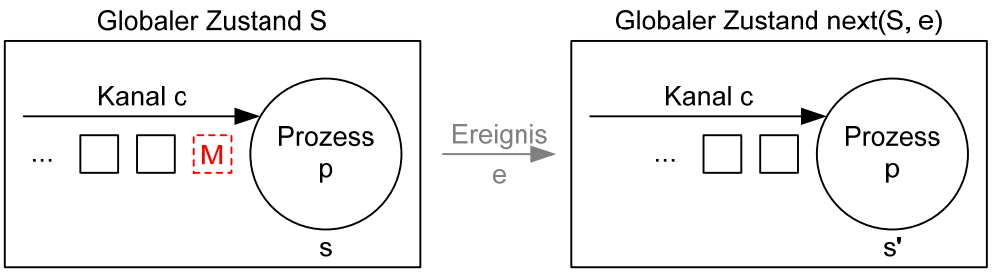

Abb. 3: Auswirkungen eines Ereignisses *e* bei einem eingehenden Kanal

Die Abfolge von *n* Ereignissen, die in allen Prozessen eines verteilten Systems auftreten, lässt sich wie folgt definieren:

$$seq = (e_i : 0 \le i \le n) \tag{2}$$

Nur wenn jedes e_i in einem globalen Zustand S_i auftreten kann, wobei S_0 den Anfangszustand beschreibt, dann beschreibt *seq* die Berechnung des verteilten Systems und es gilt folgende Rekursion:

$$S_{i+1} = next(S_i, e_i) \text{ für } 0 \le i \le n\,. \tag{3}$$

2.2.2 Konsistente Schnappschüsse

An dieser Stelle soll das Überweisungsbeispiel aus dem Unterkapitel 2.1 noch mal genau betrachtet werden (vgl. Abb. 1). Mangels einer Synchronisierung wurden bei dem Erstellen des Schnappschusses der Prozess des Kontos 1 im globalen Zustand 1, die beiden Kanäle im globalen Zustand 2 und der Prozess des Kontos 2 im globalen Zustand 3 aufgezeichnet. Die Summe aus den Kontostände (500 und 250 Euro) und dem gerade übertragenen Geld (50 Euro) ist um 150 Euro größer als sie in einem konsistenten Zustand sein dürfte. Der Grund für diese *Inkonsistenz* ist, dass der lokale Zustand des Kontos 1 aufgezeichnet wurde, bevor die Nachricht verschickt wurde, der lokale Zustand vom Kanal 1 wurde erfasst, während die Nachricht übertragen wurde, und der lokale Zustand des Kontos 2 festgehalten wurde, nachdem die Nachricht vom Konto 2 empfangen wurde. In anderen Worten heißt dies, dass der festgehaltene lokale Zustand von Konto 1 die sich im Kanal befindende Nachricht gar nicht kennt und Konto 2 bereits eine Nachricht empfangen hat, welche sich noch im Kanal befindet.

Hieraus folgt für einen Prozess *p* und einen von diesem Prozess ausgehenden Kanal *c*, dass ein aufgezeichneter Schnappschuss dann *inkonsistent* ist, wenn *n* und *n*' die Anzahl der von Prozess *p* über Kanal *c* verschickten Nachrichten darstellen, wobei *n* unmittelbar vor der Aufzeichnung des lokalen Zustandes des Prozesses *p* und *n*' unmittelbar vor der Aufzeichnung des lokalen Zustandes des Kanals *c* erfasst werden, und gilt:

$$n < n' \tag{4}$$

Darüber hinaus treten *Inkonsistenzen* auch im umgekehrten Fall auf: Würde im Überweisungsbeispiel im globalen Zustand 1 der lokale Zustand des Kontos 2 erfasst, im globalen Zustand 2 der lokale Zustand des Kontos 1 und im globalen Zustand 3 der lokale Zustand der beiden Kanäle, so würde der Schnappschuss ein Defizit von 50 Euro aufweisen. Der Grund ist, dass im festgehaltenen Zustand Konto 1 eine Nachricht verschickt hat, die sich weder in dem Zustand des Kanals 1 befindet noch von Konto 2 empfangen wurde. Dies lässt sich auch mit n und n' ausdrücken:

$$n > n' \tag{5}$$

Aus (4) und (5) folgt, dass ein globaler Zustand *konsistent* ist, wenn

$$n = n' \tag{6}$$

Nachdem das Senden über einen ausgehenden Kanal betrachtet wurde, folgt analog für das Empfangen von Nachrichten für einen Prozess q, in den Kanal c eingeht: Ein Schnappschuss ist dann *konsistent*, wenn m und m' die Anzahl der von Prozess q über Kanal c empfangenen Nachrichten darstellen, wobei m unmittelbar vor der Aufzeichnung des lokalen Zustandes des Prozesses q und m' unmittelbar vor der Aufzeichnung des lokalen Zustandes des Kanals c erfasst werden, und gilt:

$$m = m' \tag{7}$$

In keinem verteilten System kann die Anzahl der über einen Kanal empfangen Nachrichten die Anzahl der über denselben Kanal gesendeten Nachrichten überschreiten. Mit n' und m' ausgedrückt, ergibt sich folgende Bedingung:

$$n' \geq m' \tag{8}$$

Aus den Gleichungen (6), (7) und (8) folgt:

$$n \geq m \tag{9}$$

In Worten bedeutet dies, dass in einem konsistenten Schnappschuss die Anzahl der vom Prozess p gesendeten Nachrichten mindestens so groß sein muss wie die Anzahl der vom Prozess q empfangenen Nachrichten. Der lokale Zustand des Kanals c ergibt sich dann aus den festgehaltenen lokalen Zuständen der beiden Prozesse p und q. Hierbei sind zwei Fälle zu unterscheiden:

1. Stimmt vor der Aufzeichnung der lokalen Zustände der beiden Prozesse die Anzahl der über den Kanal c gesendeten und der empfangenen Nachrichten überein ($n = m$), dann gilt wegen (6) und (7) auch $n' = m'$. Da alle gesendeten Nachrichten empfangen wurden, befinden sich keine Nachrichten in der Übertragung und der Kanal c kann als leer im Schnappschuss festgehalten werden.

2. Ist dagegen vor dem Festhalten der lokalen Zustände der beiden Prozesse die Anzahl der über den Kanal *c* gesendeten Nachrichten größer als die Anzahl der empfangenen Nachrichten ($n \geq m$), dann ergibt sich im Schnappschuss der lokale Zustand des Kanals *c* als die Folge aller Nachrichten, die der Prozess *p* vor dem Erfassen des lokalen Zustands des Prozesses *p* über Kanal *c* gesendet hat, ohne die Folge aller Nachrichten, die der Prozess *q* vor dem Erfassen des lokalen Zustandes des Prozesses *q* empfangen hat.

Im Schnappschuss enthält der Kanal *c* also $n - m$ Nachrichten. Es handelt sich dabei um die $(m + 1)$-te bis *n*-te von Prozess *p* gesendeten Nachrichten. Sind in einem konsistenten Schnappschuss alle Kanäle leer, so handelt es sich um einen *streng konsistenten Schnappschuss* [SS94, S. 110]. Dann gilt statt (9) sogar:

$$n = m \tag{10}$$

Zusammenfassend zeigt folgende Abbildung die vorgestellten Schnappschuss Arten:

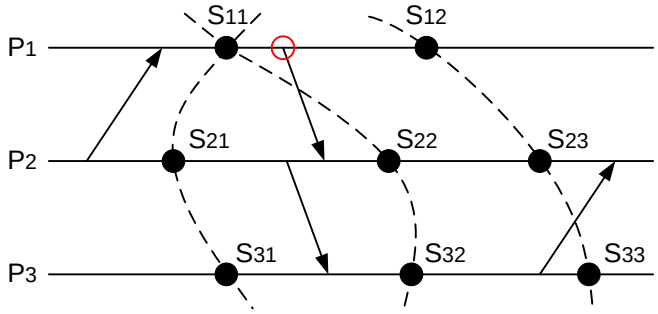

Abb. 4: Schnappschüsse im Prozess-Zeit-Diagramm (Quelle: [SS94, S. 111])

Es handelt sich hierbei um ein Prozess-Zeit-Diagramm mit drei Prozessen p_1, p_2 und p_3, deren lokale Zustände s_{ij} zu verschiedenen Zeitpunkten festgehalten wurden, wobei *i* der Index des Prozesses ist und *j* eine Nummerierung der lokalen Zustände des Prozesses p_i. Die Pfeile stellen dabei Nachrichten dar, wobei die Pfeilspitze die Richtung des Kanals andeutet. Bei dem Schnappschuss $\{s_{11}, s_{21}, s_{31}\}$ handelt es sich um einen streng konsistenten Schnappschuss, da in keinem lokalen Zustand gesendete Nachrichten, die noch nicht empfangen wurden, oder empfangene Nachrichten, deren Versendung unbekannt ist, existieren. Der Schnappschuss $\{s_{12}, s_{23}, s_{33}\}$ ist ein konsistenter – aber nicht streng konsistenter – Schnappschuss, da hier die Bedingung (10) nicht erfüllt ist: Im lokalen Zustand s_{33} ist eine versendete Nachricht enthalten, die im Zustand s_{23} noch nicht empfangen wurde. Dagegen ist Schnappschuss $\{s_{11}, s_{22}, s_{32}\}$ ein inkonsistenter Schnappschuss, weil im Zustand s_{22} eine Nachricht empfangen wurde, die im Zustand s_{11} noch nicht versendet wurde (vgl. roter Kreis in Abb. 4).

3 Festhalten von globalen Zuständen

In diesem Kapitel wird nun der Chandy-Lamport-Algorithmus vorgestellt, der die Konsistenzbedingung aus dem letzten Kapitel erfüllt. Es existierten bereits vor der Veröffentlichung des Algorithmus von CHANDY und LAMPORT im Jahre 1985 Algorithmen zur Erfassung von globalen Schnappschüssen. Bspw. ermöglichte der Algorithmus von FISCHER, GRIFFETH und LYNCH aus dem Jahre 1982 das Festhalten eines möglichen Schnappschusses eines Systems, das aus vielen unabhängigen Transaktionen besteht [FG82]. Doch wird dafür zusätzlich eine Kopie des Transaktionssystems parallel ausgeführt, welche nach der Initiierung des Algorithmus keine neuen Transaktionen mehr startet und die laufenden Transaktionen bis zu ihrem Ende ausführt, um dann einen konsistenten Zustand zu erreichen. Dagegen erfordert der Algorithmus von CHANDY und LAMPORT keine Ressourcen benötigende Kopie und kann auch die Zustände der Nachrichtenkanäle erfassen.

3.1 Chandy-Lamport Algorithmus

Um die Grundidee des Chandy-Lamport Algorithmus zu verdeutlichen, sollen ein Prozess *p*, ein Prozess *q* und ein Kanal, der von *p* zu *q* verläuft, betrachtet und das Festhalten des lokalen Zustands von *c* erläutert werden. Nachdem *p* seinen lokalen Status festgehalten hat, sendet er eine spezielle Nachricht über *c* zu *q*, bevor *p* weitere Nachrichten über den Kanal sendet. Die spezielle Nachricht wird *Marker* genannt und beeinflusst die eigentliche Berechnung des verteilten Systems nicht. Der lokale Zustand des Kanals *c* ergibt sich für den Prozess *q* als die Nachrichtenfolge, die *q* empfangen hat,

1. nachdem er seinen lokalen Zustand festgehalten hat und
2. bevor der Marker über den Kanal *c* erhalten wird.

Hat der Prozess *q* seinen lokalen Status bei Erhalt des Markers noch nicht festgehalten, so muss er dies zuerst erledigen, bevor er weitere Nachrichten verarbeitet. Da in diesem Fall zwischen der Ankunft des Markers und dem Aufzeichnen des lokalen Zustands aufgrund des Warteschlangen-Verhaltens keine weiteren Nachrichten verarbeitet werden, ist der Kanal *c* als leer aufzuzeichnen.

Dies ist die Pseudo-Code Darstellung des Chandy-Lamport Algorithmus [CL85, S. 70]:

Marker-Senderegel für einen Prozess p

```
Für jeden Kanal c, der von p abgehend ist:
p sendet einen einzigen Marker über c, nachdem p seinen Status
festgehalten hat und p weitere Nachrichten über c verschickt.
```

Marker-Empfangsregel für einen Prozess q

```
Beim Empfangen eines Markers über einen Kanal c:
Wenn q seinen Zustand noch nicht festgehalten hat dann
    Beginn  q hält seinen Zustand fest;
            q hält den Zustand von c als leere Sequenz fest
    Ende
sonst  q hält den Zustand von c als Sequenz der Nachrichten
       fest, die über c empfangen wurden, nachdem der Zustand
       von q festgehalten wurde und bevor q den Marker über c
       empfangen hat.
```

Der Algorithmus kann von einem beliebigen Prozess initiiert werden, welcher zuerst seinen eigenen lokalen Zustand speichert und vor dem Versenden weiterer Nachrichten einen Marker[1] über alle ausgehenden Kanäle verschickt.

Besitzt ein Prozess mehrere eingehende Kanäle, so muss dieser den gleichen Marker über jeden eingehenden Kanal erhalten, damit der Zustand für jeden Kanal im Schnappschuss festgehalten werden kann. Im folgenden Prozessgraph hat Prozess p_1 zwei eingehende Kanäle und empfängt den gleichen Marker über den Kanal c_3 und Kanal c_4:

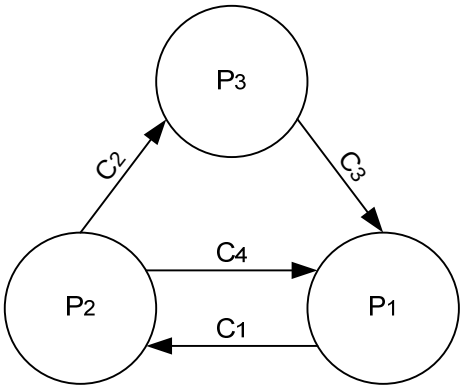

Abb. 5: Ein Prozess mit mehreren eingehenden Kanälen

Der Kanal, über den Prozess p_1 zuerst einen Marker empfängt, wird als leer aufgezeichnet. Bei dem anderen Kanal kommt es darauf an, ob p_1 seit dem Festhalten seines eigenen lokalen Zustandes Nachrichten über diesen Kanal empfangen hat.

[1] Da mehrere Schnappschuss Aufzeichnungen parallel stattfinden können, muss der Marker eine eindeutige Aufzeichnungs-Identifikation enthalten. Das könnte bspw. ein Tupel aus der Prozessnummer und einer fortlaufenden Zahl sein. So kann es sein, dass ein Prozess mehrere Schnappschüsse für einen Kanal gleichzeitig aufzeichnet.

3.1.1 Terminierung des Chandy-Lamport-Algorithmus

Damit der Alogrithmus in einer endlichen Zeit terminiert, müssen folgenden Voraussetzungen erfüllt sein [CL85, S. 70-71]:

1. Jeder Prozess muss gewährleisten, dass kein Marker endlos in einem eingehenden Kanal verbleibt.
2. Die Aufzeichnung des lokalen Zustandes eines Prozesses muss in einer endlichen Zeit erfolgen.
3. Der Prozessgraph muss *zusammenhängend* [Di06, S. 11] sein, d. h. es existiert für jeden Prozess ein Weg über beliebig viele Nachrichtenkanäle zu allen anderen Prozessen. Dies bedeutet, dass mindestens ein Weg von jedem Prozess zu sich selbst existiert.

Wenn alle genannten Vorraussetzungen erfüllt sind, dann ist sichergestellt, dass ein Marker über jeden eingehenden Kanal aller Prozesse empfangen und der lokale Zustand jedes Prozesses aufgezeichnet wird: Wenn ein Prozess *p*, ohne einen Marker zu erhalten, seinen lokalen Zustand aufgezeichnet hat, ein Nachrichtenkanal von *p* zu einem Prozess *q* besteht, dann wird *q* seinen lokalen Zustand aufgrund der ersten und der zweiten Voraussetzungen in einer endlichen Zeit festhalten. Ist nun *q* ein beliebiger Prozess, zu dem von *p* ein Weg über mehrere Nachrichtenkanäle besteht, so werden alle Prozesse auf dem Weg von *p* nach *q* ihre lokalen Zustände in einer endlichen Zeit aufzeichnen. Unter der dritten Voraussetzung kann dann *q* jeder Prozess des verteilten Systems sein, sodass eine Aufzeichnung des gesamten verteilten Systems in einer endlichen Zeit garantiert wird.

3.1.2 Verteilung der aufgezeichneten lokalen Zustände

Nachdem die lokalen Zustände eines Prozesses und der eingehenden Nachrichtenkanälen festgehalten wurden, gibt es verschiedene Möglichkeiten, was mit den Informationen geschehen kann [KR95, S. 227].

- Ist der globale Schnappschuss nur für den initiierenden Prozess von Bedeutung, so sendet jeder Prozess seinen Zustand und die Zustände der eingehenden Kanäle nur in Richtung des initiierenden Prozesses. Dies kann über den kürzesten oder kostengünstigsten Weg von Nachrichtenkanälen erfolgen.

- Ist dagegen der globale Schnappschuss nicht nur für den initiierenden Prozess, sondern für alle anderen Prozesse von Interesse, so verschickt jeder Prozess die Zustände über alle ausgehenden Kanäle[2]. Aufgrund der dritten Voraussetzung für die Terminierung des Algorithmus in endlicher Zeit empfängt mit dieser Regel jeder Prozess alle lokalen Zustände.

Eine weitere denkbare Möglichkeit wäre, die Zustände an eine Gruppe von Prozessen zu schicken. Jeder Prozess kann für sich den globalen Schnappschuss aus den empfangenen lokalen Zuständen zusammensetzen.

3.1.3 Aufwand des Chandy-Lamport Algorithmus

Für die Angabe des Aufwands des Chandy-Lamport Algorithmus wird nur der Teil betrachtet, in dem die Aufzeichnung der lokalen Zustände der Prozesse und Kanäle stattfindet. Eine einzelne Ausführung des Algorithmus benötigt $O(e)$ Nachrichten und $O(d)$ Zeit, wobei e die Anzahl der Nachrichtenkanäle – Kanten im Prozessgraph – ist und d den Durchmesser des Prozessgraphs[3] angibt [KR95, S. 227]. Der Aufwand für das Verschicken und das Zusammensetzen dieser Zustände zu einem globalen Schnappschuss hängt erstens von der gewählten Verteilungsstrategie und zweitens von dem für das Zusammensetzen verwendeten Verfahren ab.

3.2 Eigenschaften festgehaltener Schnappschüsse

Nachdem im vorherigen Unterkapitel die Vorgehensweise zur Erstellung eines Schnappschusses beschrieben wurde, geht es in diesem Kapitel nun um die Eigenschaften eines festgehaltenen Schnappschusses. Zur Verdeutlichung soll hier wieder ein Kontenbeispiel dienen, bei dem zwei Überweisungen parallel stattfinden (vgl. Abb. 6):

---

[2] Dies entspricht einer Überflutung eines Netzwerkes (engl. flooding). Existieren von einem Prozess, zwei unterschiedliche Wege über denselben nachgelagerten Prozess, so wird der Zustand nur einmal vom nachgelagerten Prozess weiter verschickt.

[3] Der Durchmesser eines Graphen gibt den größten Abstand zweier beliebiger Ecken in einem Graphen an, wobei unter Abstand der kürzeste Weg zwischen zwei Ecken verstanden wird [Di06, S. 9].

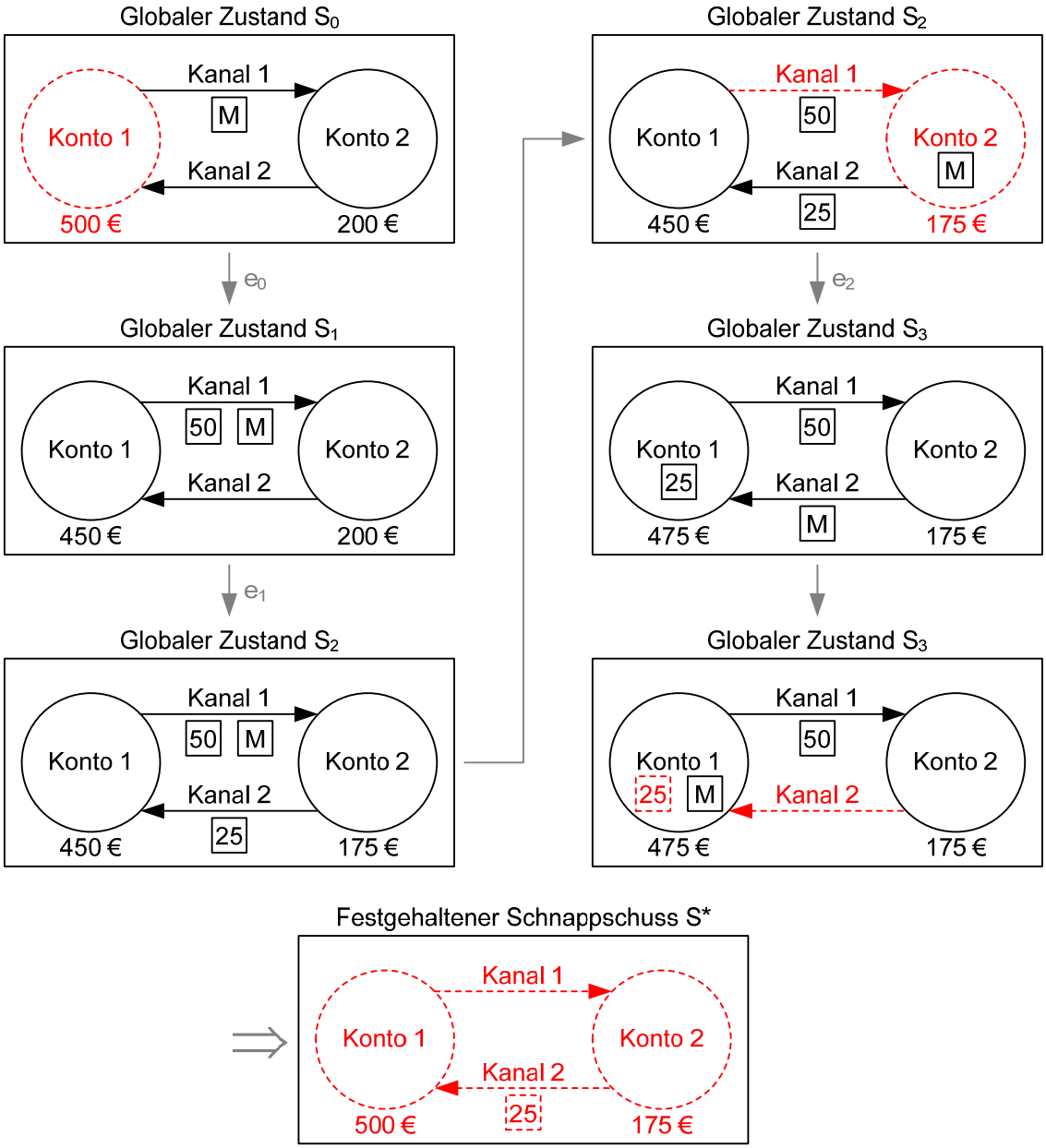

Abb. 6: Festgehaltener Schnappschusses entspricht keinem globalen Zustand

Im Anfangszustand S_0 beträgt der Kontostand von Konto 1 500 Euro und von Konto 2 200 Euro. In diesem globalen Zustand initiiert Konto 1 die Erstellung eines Schnappschusses, hält seinen lokalen Zustand fest und versendet einen Marker – hier mit M bezeichnet – über Kanal 1, bevor weitere Nachrichten verschickt werden. Durch das Ereignis $e_0 = \langle Konto\,1, 500\,€, 450\,€, 50\,€, Kanal\,1 \rangle$ wechselt das System in den globalen Zustand S_1, in dem eine Überweisung von 50 Euro von Konto 1 zu Konto 2 in der Nachrichten Warteschlange hinter dem Marker eingereiht ist. Als nächstes schickt Konto 2 mit dem Ereignis $e_1 = \langle Konto\,2, 200\,€, 175\,€, 25\,€, Kanal\,2 \rangle$ eine Überweisung von 25 Euro zu Konto 1 ab und der globale Zustand S_2 wird erreicht. Nun nimmt Konto 2

den Marker von dem Kanal 1 entgegen, hält seinen lokalen Zustand als 175 Euro und den Zustand des eingehenden Kanals 1 als leer fest, da Konto 2 zum ersten Mal den Marker *M* empfängt. Da dieser Vorgang die Berechnung des verteilten Systems nicht beeinflusst, wird dieser globale Zustand wieder mit S_2 bezeichnet. Jetzt tritt das Ereignis $e_2 = \langle Konto\ 1, 450\,€, 475\,€, 25\,€, Kanal\ 2 \rangle$ ein und das System wechselt in den globalen Zustand S_3. Außerdem wird über Kanal 2 der Marker in Richtung Konto 1 verschickt. Dies ist nicht als einzelner Schritt dargestellt, da es weder die Berechnung beeinflusst noch dabei ein Teil des Systems festgehalten wird. Schließlich empfängt Konto 1 über Kanal 2 der Marker *M* und hält den Zustand von Kanal 2 fest, welcher aus der im Ereignis e_2 empfangenen Nachricht – die Überweisung von 25 Euro – besteht, weil dies die einzige empfangene Nachricht ist, seitdem Konto 1 die Aufzeichnung des Schnappschusses begonnen hat.

3.2.1 Lokale Zustände zu einem globalen Schnappschuss zusammensetzen

Setzt man die einzelnen festgehaltenen lokalen Zustände zu einem globalen Schnappschuss *S** zusammen, so fällt auf, dass dieser keinem der jemals eingetretenen globalen Zustände entspricht[4]. Nun stellt sich die Frage, ob dieser Schnappschuss überhaupt Sinn und Zweck hat. Um diese Frage zu beantworten, soll herausgefunden werden, ob nach dem globalen Zustand, der bei dem Start des Chandy-Lamport-Algorithmus besteht, ein globaler Zustand erreicht werden kann, der dem Schnappschuss *S** entspricht, und ob von diesem Zustand dann der globale Zustand erreicht werden kann, der bei der Terminierung des Algorithmus existiert.

Hierzu wird eine allgemeine verteilte Berechnung betrachtet, welche durch $seq = (e_i, 0 \le i)$ dargestellt wird (vgl. Kapitel 2.2.1) [CL85, S. 71 ff]. Dann ist S_i der globale Zustand, der genau kurz vor dem Eintreten des Ereignisses $e_i, 0 \le i$ aus *seq* erreicht wird. Der Algorithmus wird im globalen Zustand S_ι initiiert und terminiert im Zustand $S_\phi, 0 \le \iota \le \phi$. Es ist nun Folgendes nachzuweisen:

1. *S** ist erreichbar von S_ι und
2. S_ϕ ist erreichbar von *S**.

---

[4] Beim Vergleich des festgehaltenen Schnappschusses *S** mit den globalen Zuständen S_0, …, S_3 ist der Marker *M* zu ignorieren, da dieser kein Teil der Berechnung ist.

Dafür müssen wir zeigen, dass eine Berechnung *seq*' existiert, sodass:

1. *seq*' eine Permutation von *seq* ist, in welcher die globalen Zustände *S**, S_ι und S_ϕ auftreten,
2. $S_\iota = S^*$ ist oder S_ι früher als *S** eintritt und
3. $S_\phi = S^*$ ist oder *S** früher als S_ϕ in *seq*' eintritt.

Behauptung: Es existiert eine verteilte Berechnung $seq' = (e_i', 0 \le i)$, für die gilt:

1. Für alle *i*, bei denen $i < \iota$ oder $i \ge \phi : e_i' = e_i$,
2. die Teilsequenz $(e_i', \iota \le i < \phi)$ ist eine Permutation der Teilsequenz $(e_i, \iota \le i < \phi)$,
3. für alle *i*, bei denen $i < \iota$ oder $i \ge \phi : S_i' = S_i$ und
4. es existiert ein *k*, sodass $\iota \le k < \phi$ und $S^* = S_k'$.

3.2.2 Beweis der Behauptung, dass der festgehaltene Schnappschuss einen erreichbaren globalen Zustand darstellt

Die einzelnen Ereignisse der Sequenz *seq* werden in zwei Gruppen eingeteilt: Findet ein Ereignis e_i in einem Prozess *p* statt und wird der lokale Zustand von *p* erst nach dem Eintreten von e_i festgehalten, so ist e_i ein *vor Aufzeichnung stattfindendes Ereignis*. Analog ist e_i ein *nach Aufzeichnung stattfindendes Ereignis*, wenn der lokale Zustand von *p* erst nach dem Eintreten von e_i festgehalten wird. Hieraus folgt, dass alle Ereignisse $e_i, i < \iota$ vor Aufzeichnung stattfindende und $e_i, i \ge \phi$ nach Aufzeichnung stattfindende Ereignisse sind. Dagegen kann ein nach Aufzeichnung stattfindendes Ereignis e_{j-1} vor einem vor Aufzeichnung stattfindenden Ereignis e_j mit $\iota < j < \phi$ eintreten. Voraussetzung hierfür ist, dass sich e_{j-1} und e_j auf verschiedenen Prozessen ereignen, da sonst die Aufzeichnung des lokalen Zustands desselben Prozesses bereits in e_{j-1} erfolgt wäre.

In dem Beispiel in Abb. 6 ist $S_\iota = S_0$ und $S_\phi = S_3$, d. h. die gesamte Abbildung zeigt nur den Teil der Berechnung, der während des Ablaufs des Algorithmus ausgeführt wird. Beim Betrachten der Ereignisse e_0, e_1 und e_2 fällt auf, dass e_0 nach der Aufzeichnung des Zustands von Konto 1 eintritt und e_1 und e_2 vor der Aufzeichnung des Kontos 2 bzw. des Kanals 2 eintreten. Veranschaulicht wird dies in folgender Abbildung:

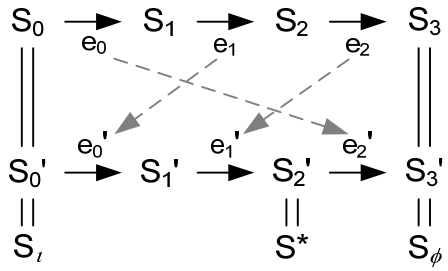

Abb. 7: Permutation der vor Aufzeichnung und nach Aufzeichnung stattfindenden Ereignisse

Eine Verschiebung des Ereignisses e_0, welches nach der Aufzeichnung stattfindet, führt dazu, dass im globalen Zustand S_2' ein Zustand erreicht wird, der dem festgehaltenen Zustand S* entspricht.

Die Erklärung erfolgt in zwei Schritten:

1. Zeigen, dass eine Permutation der Ereignisse erlaubt ist, sodass alle vor Aufzeichnung stattfindenden Ereignisse vor den nach Aufzeichnung stattfindenden eintreten.
2. Beweisen, dass der globale Zustand dem festgehaltenen Zustand entspricht, nachdem alle vor Aufzeichnung stattfindenden Ereignisse eingetreten sind.

Beim ersten Schritt wird davon ausgegangen, dass Ereignis e_{j-1} im Prozess p und e_j im Prozess q stattfindet. Eine Permutation der Ereignisse e_{j-1} und e_j erfordert nun, dass e_{j-1} nicht den lokalen Zustand von q betrifft, d. h. in e_j eine Nachricht empfangen wird, die in e_{j-1} verschickt wurde. Dies kann nicht der Fall sein, wenn e_{j-1} ein nach Aufzeichnung stattfindendes Ereignis ist, weil aufgrund der Warteschlangen-Eigenschaft des Kanals zwischen den Prozessen p und q erstens p vor dem Versenden der Nachricht einen Marker verschickt haben müsste und zweitens q diesen Marker vor der verschickten Nachricht empfangen würde. Also wird der lokale Zustand des Prozesses q nicht durch das Ereignis e_{j-1} verändert und ein Eintreten der Ereignisse e_1, …, e_{j-2}, e_j, e_{j-1} führt zu demselben globalen Zustand wie e_1, …, e_{j-2}, e_{j-1}, e_j. Dieses Vertauschen zweier Ereignisse wird solange durchgeführt, bis eine Permutation erreicht wird, in der alle vor Aufzeichnung stattfindenden Ereignisse vor den nach Aufzeichnung stattfindenden eintreten.

In dem zweiten Schritt muss sowohl gezeigt werden, dass für jeden Prozess p des erstellten Schnappschusses S* der lokale Zustand nur von den vor Aufzeichnung in p stattfindenden Ereignissen abhängig ist, als auch, dass sich der lokale Zustand jedes Kanals c zwischen zwei Prozessen p und q als die Nachrichtenfolge ergibt, die p in vor Aufzeichnung stattfindenden Ereignissen versendet hat, abzüglich der Nachrichtenfolge, die aus den von q in vor Aufzeichnung stattfindenden Ereignissen empfangen hat. Die

erste Forderung ist erfüllt, da dies der Definition von vor Aufzeichnung stattfindenden Ereignissen entspricht. Zur zweiten Forderung: Der lokale Zustand des Kanals *c* ergibt sich als die Nachrichtenfolge, die Prozess *q* empfängt, nachdem *q* seinen lokalen Zustand festgehalten hat und bevor *q* einen Marker über *c* empfängt (vgl. Kapitel 2.2.2). Die Nachrichten, die von *q* über *c* empfangen werden, bevor ein Marker empfangen wird, entsprechen den Nachrichten, die durch in *p* vor Aufzeichnung stattfindenden Ereignissen über *c* verschickt werden. Außerdem ergibt sich die Menge an Nachrichten, die *q* vor der Aufzeichnung seines lokalen Zustandes empfangen hat aus den in *q* vor Aufzeichnung stattfindenden Ereignissen. Somit ist auch diese Forderung erfüllt und der globale Zustand nach allen vor Aufzeichnung stattfindenden Ereignissen entspricht dem festgehaltenen Schnappschuss *S**.

Damit ist der Beweis abgeschlossen, dass ein festgehaltener Schnappschuss – auch wenn er eventuell keinem globalen Zustand entspricht – einen in der Berechnung erreichbaren globalen Zustand beschreibt, und somit hat ein zusammengesetzter Schnappschuss durchaus Sinn und Zweck.

3.2.3 Gültigkeit globaler Prädikate in Schnappschüssen

Ein globales Prädikat hat die Eigenschaft, dass es bei der Terminierung des Algorithmus gelten muss, wenn es bei der Initiierung des Algorithmus gilt. Formal lässt sich dieser Sachverhalt folgendermaßen beschreiben [CL85, S. 74]:

$$y(S_\iota) \rightarrow \textit{endgültig} \text{ und } \textit{endgültig} \rightarrow y(S_\phi) \tag{11}$$

Hierbei ist $y(S)$ ein globales Prädikat im globalen Zustand *S*, *endgültig* eine boolesche Variable und „$\rightarrow$“ stellt die Implikation der Prädikatenlogik dar. Es besteht also durchaus die Möglichkeit, dass sich der Wert eines globalen Prädikats während der Ausführung des Algorithmus auf *wahr* verändert ($y(S_\phi) = \textit{wahr}$), wenn der Wert bei der Initiierung *falsch* war ($y(S_\iota) = \textit{falsch}$).

Für ein globales Prädikat *y*(*S**) in einem festgehaltenen Schnappschuss *S** gilt nun:

$$y(S_\iota) \rightarrow y(S^*) \rightarrow y(S_\phi) \tag{12}$$

Ein globales Prädikat, das im festgehaltenen Zustand gilt, gilt auch nach Terminierung des Chandy-Lamport Algorithmus. Die Begründung ist, dass *S** von S_ι erreichbar ist und S_ϕ von *S** erreichbar ist.

3.3 Abweichende Annahmen bezüglich der Nachrichtenkanäle

Im diesem Kapitel werden Modifikation des Chandy-Lamport Algorithmus vorgestellt, wenn Abweichungen von Annahmen aus Kapitel 2.2.1 an die Nachrichtenkanäle bestehen. Hierfür wird nun davon ausgegangen, dass die Empfangsreihenfolge von verschiedenen Nachrichten nicht mehr nach deren Absendezeitpunkten geordnet ist, d. h. es liegen keine FiFo Warteschlagen vor. Dies hat zur Folge, dass sich Nachrichten überholen können. So kann es passieren, dass ein Marker, welcher auf Kommunikationsebene auch nur eine Nachricht darstellt, von einer Nachricht *M* überholt wird. Dadurch tritt eine Inkonsistenz auf, weil das Empfangen von *M* im Schnappschuss enthalten ist, aber das Senden von *M* nicht erfasst wurde. Dies wird im linken Teil der folgenden Abbildung deutlich:

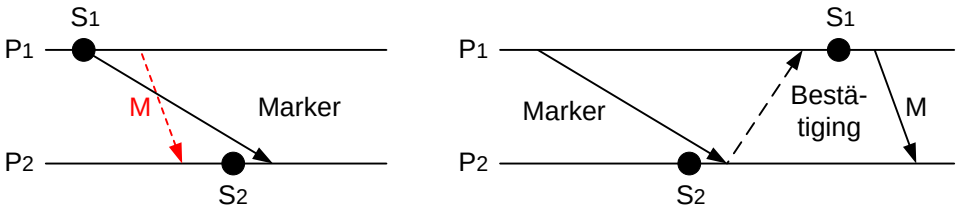

Abb. 8: Inkonsistenz im Schnappschuss bei nicht FiFo Nachrichtenkanälen (in Anlehnung an [He89, S. 8-9])

In [He89] beschreibt HELARY eine Möglichkeit, wie die Konsistenz des festgehaltenen Schnappschusses dennoch sichergestellt werden kann. In Abb. 8 beschließt ein Prozess p_1, seinen lokalen Zustand aufzuzeichnen. Vor dem Aufzeichnen des lokalen Zustandes verschickt p_1 einen Marker in Richtung des Prozesses p_2 und wartet auf eine Bestätigung[5] des Empfangs und der Aufzeichnung des lokalen Zustandes von p_2. Hierdurch zögert p_1 die Aufzeichnung seines Zustandes und vor allem das Verschicken weiterer für die Berechnung relevanter Nachrichten hinaus, sodass eine nach Aufzeichnung verschickte Nachricht nicht im erfassten globalen Schnappschuss enthalten sein kann.

Eine andere Möglichkeit, mit nicht FiFo Nachrichtenkanälen umzugehen, bietet der Lai-Yang Algorithmus [KR95, S. 229]. LAI und YANG verwenden keine Marker, sondern setzen Markierungen in den Nachrichten der Berechnung. Zur Beschreibung werden die Farben Weiß und Rot verwendet:

[5] In dem Modell des verteilten Systems wird nur auf logischer Ebene von gerichteten Nachrichtenkanälen ausgegangen, sodass es physikalisch möglich ist, Bestätigungen über einen Nachrichtenkanal zu erhalten.

- Jeder Prozess ist nach Initialisierung weiß und wechselt auf rot, wenn der lokale Zustand des Prozesses aufgezeichnet wird.
- Jede Nachricht nimmt die Farbe des Prozesses an, d. h. weiße Nachrichten wurden vor der Aufzeichnung und rote Nachrichten nach der Aufzeichnung des lokalen Zustandes verschickt.
- Jeder Prozess kann selbst entscheiden, wann der lokale Zustand aufgezeichnet wird. Dennoch muss ein Prozess seinen lokalen Zustand festhalten, bevor er eine rote Nachricht verarbeitet.

Durch dieses Vorgehen ist sichergestellt, dass kein Prozess eine Nachricht verarbeitet und mit in seinen lokalen Zustand übernimmt, die nach dem Festhalten des Status eines vorgelagerten Prozesses gesendet wurde. So können keine Inkonsistenzen auftreten. Die Farbe übernimmt also die Funktion eines Markers.

Um die lokalen Zustände der Kanäle zu ermitteln, zeichnet jeder Prozess über alle eingehenden und ausgehenden Kanäle empfangenen bzw. gesendeten weißen Nachrichten auf. Wechselt ein Prozess seine Farbe von weiß auf rot, dann sendet dieser Prozess alle Nachrichten an den initiierenden Prozess. Dieser ermittelt dann die lokalen Zustände aller Kanäle auf folgende Weise: Der Zustand eines Kanals von einem Prozess p_1 zu einem Prozess p_2 ergibt sich als Menge aller von p_1 gesendeten weißen Nachrichten abzüglich der von p_2 empfangenen Nachrichten. Da die Menge der gespeicherten weißen Nachrichten groß werden kann, handelt es sich hierbei um einen Nachteil dieses Algorithmus.

4 Zusammenfassung

Nach einer kurzen Einführung in die Geschichte verteilter Systeme (Kapitel 1), folgte zunächst im Unterkapitel 2.1 die Motivation für globale Schnappschüsse, welche durch ein Einführungsbeispiel unterstützt wurde, welches in abgewandelter Form in der gesamten Ausarbeitung wieder aufgegriffen wurde. Im Unterkapitel 2.2 wurde als Nächstes das dieser Ausarbeitung zugrundeliegende verteilte System modelliert. Das eingeführte Modell beinhaltet beliebig viele Prozesse, welche über gerichtete Nachrichtenkanäle miteinander kommunizieren. Dazu wurde gezeigt, wie das verteilte System durch Ereignisse von einem Zustand in den nächsten übergeht. Das Unterkapitel endete mit den für konsistente Schnappschüsse erforderlichen Voraussetzungen. Im Hauptkapitel 3 ging es nun um die Erstellung von konsistenten Schnappschüssen. Hierzu wurde im Unterkapitel 3.1 der Chandy-Lamport Algorithmus vorgestellt, welcher auf dem zuvor beschriebenen Modell basiert. Da ein festgehaltener Schnappschuss keinem eingetretenen globalen Zustand entsprechen muss, wurde im Unterkapitel 3.2 gezeigt, dass ein mit dem Chandy-Lamport-Algorithmus erzeugter Schnappschuss einen erreichbaren globalen Zustand darstellt, von dem wiederum der globale Zustand erreichbar ist, der bei der Terminierung des Algorithmus besteht. Hieraus lässt sich auch herleiten, dass globale Prädikate, die im Schnappschuss gelten auch nach der Terminierung gelten müssen. Schließlich wurden im Unterkapitel 3.3 zwei Algorithmen kurz vorgestellt, die das Erstellen von globalen Schnappschüssen dennoch ermöglichen, wenn die Annahme an die Nachrichtenkanäle nicht mehr gilt, dass die Empfangsreihenfolge der Sendereihenfolge entspricht.

Algorithmen zur Erfassung von globalen Schnappschüssen sind aus verteilten Systemen wie bspw. Kontosystemen für Banken nicht mehr wegzudenken. Sie dienen u. a. zur Ermittlung von globalen Prädikaten, Erzeugung von Rücksetzungspunkten und Testen von Anwendungen. Der Algorithmus von CHANDY und LAMPORT ermöglicht das Festhalten von globalen Zuständen ohne dabei die eigentliche Berechnung des verteilten Systems zu beeinflussen. Darüber hinaus gibt es viele Modifikationen und Erweiterungen dieses Algorithmus. Hierunter fallen Algorithmen, die für spezielle Zwecke wie bspw. wiederholte oder parallele Ausführung optimiert sind oder eine Lösung für veränderte Annahmen bieten.

Literaturverzeichnis

[CL85] K. Mani Chandy, Leslie Lamport: *Distributed Snapshots: Determining Global States of Distributed Systems*, ACM Transactions on Computer Systems 3(1), S. 63-75, ACM Press, 1985.

[Di06] Reinhard Diestel: *Graphentheorie*, 3. Aufl., Springer, 2006.

[FG82] Michael J. Fischer, Nancy D. Griffeth, Nancy A. Lynch: *Global States of a Distributed System*, IEEE Transactions on Software Engineering SE-8(3), S. 198-202, IEEE, 1982.

[He89] Jean-Michel Helary, *Observing global states of asynchronous distributed Applications*, INRIA-Rocquencourt, 1989.

[KR95] Ajay D. Kshemkalyani, Michel Raynal, Mukesh Singhal: *An introduction to snapshot algorithms in distributed computing*, Distributed Systems Engineering 2(4), S. 224-233, Institute of Physics Publishing, 1995.

[SS94] Mukesh Singhal, Niranjan Shivaratri: *Advanced concepts in operating systems*, McGraw-Hill, 1994.

[Ta03] Andrew S. Tanenbaum, *Verteilte Systeme. Grundlagen und Paradigmen*, Pearson Studium, 2003.