**Institut für Betriebssysteme und Rechnerverbund**

# Übungen zur Vorlesung "Verteilte Systeme", WS 02/03

http://www.ibr.cs.tu-bs.de/lehre/ws0203/vs/

Dozent: Prof. Dr. Stefan Fischer <fischer@ibr.cs.tu-bs.de> · Übungsleiter: Frank Strauß <strauss@ibr.cs.tu-bs.de>

## 8 Zeit und Nebenläufigkeit — Übung am 15.01.2003

### 8.1 Optimierung des Bully-Algorithmus

In der Vorlesung haben wir gelernt, dass Auswahlalgorithmen dazu dienen, gegenüber einer statischen Bestimmung des Koordinators eine höhere Fehlertoleranz im Falle des Ausfalls eines Koordinators zu erzielen. Wir wollen nun den Kommunikationsaufwand des Bully-Algorithmus (Folie 8-51) betrachten und ihn durch Verwendung von Broadcast-Kommunikation (bzw. Multicast) optimieren.

Wir betrachten den Worst-Case, der Eintritt, wenn in einem Netz mit $n$ Prozessen $S_1..S_n$ der Prozess ____ ausfällt und dies vom Prozess ____ zuerst bemerkt wird. Dieser schickt also Panik-Meldungen (ELECTION) an die Prozesse _____. Die Anzahlen aller gesendeten Panik-Meldungen, Replies (OK) und Master-Bekanntmachungen (COORDINATOR) tragen wir in der folgenden Tabelle für die ersten drei und die letzten zwei der $n$ Prozesse ein.

| Prozess | ELECTIONs | OKs | COORDINATORs |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| ... | ... | ... | ... |
| | | | |
| | | | |

Die gesamte Anzahl der zu verschickenden Nachrichten ergibt sich also aus folgender Formel:

---

Der Aufwand liegt also in der Größenordnung $O(\_\_)$.

Wenn ein Broadcast- oder Multicast-Mechanismus zur Verfügung steht, lässt sich dieser Aufwand durch die folgende Variante des Bully-Algorithmus deutlich verringern:

- Ein Prozess $S_i$ bemerkt den Ausfall des alten Koordinationsservers und startet den Algorithmus: Er schickt einen ELECTION-Broadcast und wartet anschließend ein Zeitintervall $T$ lang auf Antwort.
- Jeder Prozess $S_j$, der eine ELECTION-Meldung von $S_i$ mit $i < j$ erhält, schickt ein OK-Broadcast, um alle $S_k$ mit $k < j$ zu beruhigen und wartet anschließend ein Zeitintervall $T$ lang auf Antwort.
- Erhält ein Prozess $S_k$ ein Zeitintervall $T$ nach Senden eines ELECTION- oder OK-Brodcasts kein OK von einem $S_j$ mit $j > k$, bestimmt sich $S_k$ zum neuen Koordinationsserver und benachrichtigt alle Prozesse mittels eines entsprechenden COORDINATOR-Broadcasts.

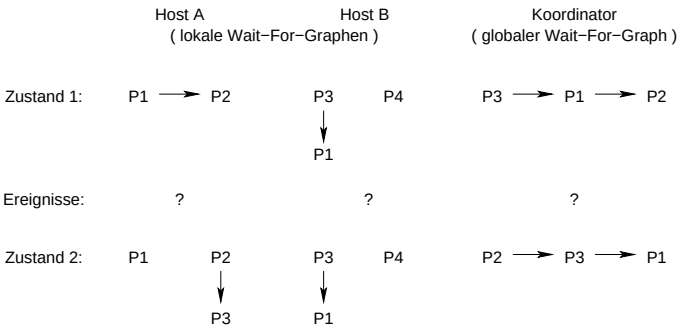
- Erhält ein Prozess $S_k$ ein OK von einem $S_j$ mit $j > k$, wartet er ein weiteres Zeitintervall $T'$ auf die Bestätigung, dass ein Prozess $S_j$ mit $j > k$ neuer Koordinationsserver ist. Trifft innerhalb von $T'$ keine Antwort ein, wird der Algorithmus neu gestartet.

Der Aufwand reduziert sich also im Worst-Case auf ___ Broadcasts (___ × ELECTION, ___ × OK, ___ × COORDINATOR) und somit auf die Größenordnung $O(\_\_\_)$.

# 9 Transaktionen

## 9.1 Erkennen von Verklemmungen

Da in verteilten Systemen das Verhindern bzw. Vermeiden von Verklemmungen deutliche Effizienzprobleme aufweist, greift man meist auf das *Erkennen von Verklemmungen* zurück. Hierzu untersucht man den globalen Wait-For-Graphen auf Zyklen. Dies kann durch einen zentralen Ansatz geschehen, bei dem Veränderungen der lokalen Wait-For-Graphen an den Koordinator gemeldet werden, so dass dieser den globalen Wait-For-Graphen aktualisieren und auf Zyklen untersuchen kann. Betrachten Sie die folgende Abbildung:



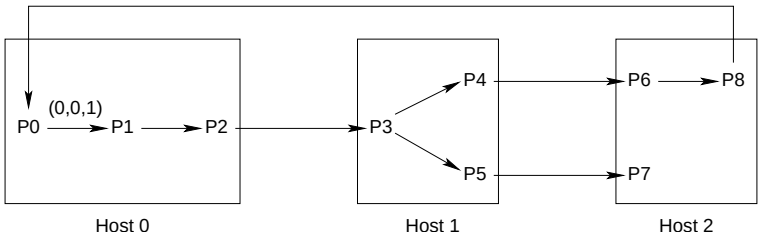

- Welche Ereignisse sind offensichtlich zwischen Zustand 1 und Zustand 2 eingetreten?
- Unter welchen Umständen kann es passieren, dass der Koordinator hier eine Verklemmung erkennt, obwohl keine Verklemmung besteht?

## 9.2 Verteiltes Erkennen von Verklemmungen

Statt eines zentralen Ansatzes, ist es auch möglich, Zyklen im globalen Wait-For-Graphen durch "edge-chasing" Nachrichten zu erkennen. Wir betrachten hier das Prinzip des Algorithmus von Chandy-Misra-Haas:

Der Algorithmus setzt ein, sobald ein Prozess auf die Betriebsmittelfreigabe eines anderen Prozesses warten muss: Es wird eine *Probe-Nachricht* entlang des Wait-For-Graphen, also an den Prozess, auf den gewartet wird, losgeschickt. Die Probe-Nachricht besteht aus einem Tripel von Prozessidentifikationen: $probe = (P_{originator}, P_{sender}, P_{receiver})$. Betrachten Sie das folgende Beispiel. Wir nehmen an, dass im

derzeitigen Zustand $P_0$ ein Betriebsmittel anfordert, dass derzeit von $P_1$ belegt wird. Die anderen Wartesituationen bestehen bereits.




- Vervollständigen Sie die Abbildung, indem Sie die Tripel aller folgenden Probe-Nachrichten einzeichnen.
- An welcher Stelle und durch welche Umstände kann hier die neu entstandene Verklemmungssituation erkannt werden?
- Durch welche Umstände könnte hier eine nicht existierende Verklemmungssituation (Phantom-Deadlock) erkannt werden?
- Welche Maßnahmen könnten eingeleitet werden, wenn eine Verklemmungssituation erkannt wurde?