UNIVERSITÄT SIEGEN

Fakultät IV
Betriebssysteme und verteilte Systeme
Prof. Dr. rer. nat. Roland Wismüller

Aufgabenblatt 8

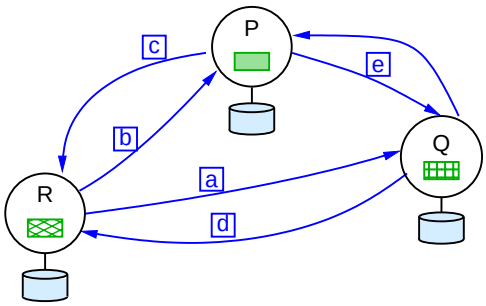
(Zu bearbeiten bis 12.12.)

Vorlesung Verteilte Systeme Wintersemester 2024/25

Aufgabe 1: Schnappschuß-Algorithmus nach Chandy/Lamport (Pflichtaufgabe für 6 LP, Abgabe über moodle!)

In der Vorlesung (Kap. 6.4) haben Sie gelernt, wie nach Chandy und Lamport ein konsistenter Zustand in einem verteilten System ermittelt wird.

Gegeben sei ein verteiltes System mit drei Prozessen P, Q, und R, die über zwei unidirektionale Kanäle verbunden sind. Das System befinde sich in dem in der Abbildung gezeigten Ausgangszustand. Die an den Kommunikationsverbindungen angegebenen Nachrichten befinden sich in Übertragung, d.h. sind gesendet, aber noch nicht empfangen.



a) Erklären Sie die Arbeitsweise des Algorithmus in diesem Fall und geben Sie den von ihm zurückgegebenen globalen Zustand an (Hinweis: dieser Zustand ist nicht eindeutig bestimmt!). Nehmen Sie an, dass Prozess Q den Schnappschuss-Algorithmus initiiert.

b) Zeichnen Sie ein Ablaufdiagramm der Prozesse und geben Sie die Schnitte für die folgenden globalen Zustände an:

- den dargestellen Ausgangszustand
- den vom Schnappschuss-Algorithmus bestimmten Zustand

Aufgabe 2: Schnappschuß-Algorithmus nach Chandy/Lamport

Zwei Prozesse P und Q sind unter Verwendung von zwei Kanälen in einem Ring verbunden, und sie schicken ständig eine Nachricht m zwischen sich hin und her. Es gibt zu jedem Zeitpunkt nur eine Kopie von m im System. Der Zustand jedes Prozesses besteht aus der Anzahl, wie oft er m bereits empfangen hat. P fängt an m zu senden. Zu einem bestimmten Zeitpunkt hat P die Nachricht und sein Zustand ist 101. Unmittelbar nach dem Senden von m initiiert P den Schnappschuss-Algorithmus.

Führen Sie der Algorithmus in diesem Fall aus und geben Sie den von ihm zurückgegebenen globalen Zustand an.

Aufgabe 3: Programmierung: Schnappschuß-Algorithmus nach Chandy/Lamport (Pflichtaufgabe für 6 LP, Abgabe über moodle!)

Im Archiv `u08Files.zip`[1] auf der Vorlesungswebseite finden Sie die Realisierung eines Prozeß-Systems mit mehreren Server-Prozessen. Die Server erhalten über Nachrichten Aufgaben, die sie lokal bearbeiten und an andere Server weitergeben können. Zusätzlich ist ein Client vorhanden, der Aufgaben an einen der Server sendet.

Erweitern Sie den Server-Code um eine Implementierung des Schnappschuß-Algorithmus nach Chandy und Lamport, so daß jederzeit die Gesamtzahl der Aufgaben im System ermittelt werden kann. Wie in der Vorlesung besprochen, besteht die Schwierigkeit darin, daß einige Aufträge zwischen zwei Servern „unterwegs“ sein können und mitgezählt werden müssen. Damit Sie den Algorithmus einfacher übertragen können, basiert das Programm auf einem klassischen Nachrichtenaustausch über Kanäle, der hier über eine RMI-Methode `enqueue()` zum Anfügen einer Nachricht an die Warteschlange des Empfängers nachgebildet wurde. Um das Ergebnis des Schnappschusses leicht überprüfen zu können, werden die Aufträge im System nie beendet (sondern nur immer von einem Server zum anderen weitergereicht), so daß die Gesamtzahl der Aufträge sich nicht ändert.

Nach Beendigung eines Schnappschusses soll jeder Server den von ihm ermittelten Zustand (d.h. die von ihm anteilig ermittelte Anzahl der Tasks) selbst ausgeben. Eine Aufsammlung der lokalen Zustände (d.h. das Zusammenrechnen der Gesamtzahl der Aufträge) müssen Sie nicht programmieren. Modifizieren Sie nur die Datei `Server.java`.

Zum Initiieren eines Schnappschusses ist ein `SnapshotClient` vorgegeben, der eine Marker-Nachricht an einen der Server sendet. Testen Sie Ihre Lösung mit unterschiedlich vielen Aufträgen und unterschiedlich vielen Servern.

Aufgabe 4: Bully-Algorithmus

In der Vorlesung wurde der Bully-Algorithmus vorgestellt (s. Kap. 7.1). Diskutieren Sie dazu folgende Fragestellungen:

a) Angenommen zwei Prozesse erkennen gleichzeitig, dass der Koordinator ausgefallen ist, und halten beide eine Wahl unter Verwendung des Bully-Algorithmus ab. Was passiert?

b) Im Bully-Algorithmus startet ein wiederhergestellter Prozess eine Wahl und wird zum neuen Koordinator, wenn er eine höhere ID hat als der aktuelle Inhaber. Ist dies ein notwendiges Funktionsmerkmal des Algorithmus?

c) Schlagen Sie vor, wie der Bully-Algorithmus angepasst werden kann, so dass er mit einer temporären Unterbrechung des Netzwerks (langsame Kommunikation) und ebenso langsamen Prozessen zurecht kommt.

Aufgabe 5: Ring-Algorithmus

In der Vorlesung (Kap. 7.1) war bei den Wahl-Algorithmen ein Ring-Algorithmus dargestellt. Die Animation im Skript zeigt zwei ELECTION-Nachrichten, die gleichzeitig kreisen. Es ist zwar kein Problem, wenn es zwei solcher Nachrichten gibt, aber es wäre eleganter, wenn eine davon entfernt werden könnte.

Entwickeln Sie einen Algorithmus, der genau dies tut, ohne die Arbeitsweise des grundlegenden Wahl-Algorithmus zu stören.

Aufgabe 6: Zentralisierter Algorithmus zum wechselseitigen Ausschluß

a) Im zentralisierten Ansatz des wechselseitigen Auschlusses (s. Vorlesung Kap. 7.2) erteilt der Koordinator nach Empfang einer Nachricht von einem Prozess, der seinen exklusiven Zugriff auf den kritischen Bereich freigibt, normalerweise dem ersten Prozess in der Warteschlange die Berechtigung. Nennen Sie einen anderen möglichen Algorithmus für den Koordinator.

b) Angenommen, der Koordinator stürzt ab. Ist damit auch immer das gesamte System arbeitsunfähig? Falls nicht, unter welchen Umständen passiert dies? Gibt es eine Möglichkeit, das Problem zu vermeiden und das System in die Lage zu versetzen, Abstürze des Koordinators zu kompensieren?

[1] `http://www.bs.informatik.uni-siegen.de/web/wismueller/vl/vs/u08Files.zip`