UNIVERSITÄT SIEGEN

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

Aufgabenblatt 9

(Zu bearbeiten bis 26.06.2020)

Vorlesung Verteilte Systeme Sommersemester 2020

Aufgabe 1: 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?

Aufgabe 2: Algorithmus von Ricart und Agrawala

Der Algorithmus von Ricart und Agrawala hat das Problem, dass falls ein Prozess abgestürzt ist und nicht auf eine Anforderung von einem anderen Prozess reagiert, in einem kritischen Bereich einzutreten, die fehlende Antwort als Verweigerung der Berechtigung interpretiert wird. Eine Methode, um abgestürtzte Prozesse einfach zu erkennen, besteht darin, alle Anforderungen unmittelbar (ggf. negativ) zu beantworten. Gibt es Situationen, wo selbst diese Methode nicht ausreichend ist?

Aufgabe 3: Deadlocks - Ricart und Agrawala

Ein verteiltes System kann mehrere voneinander unabhängige kritische Bereiche haben. Angenommen, Prozess 0 will in den kritischen Bereich A und Prozess 1 will in den kritischen Bereich B eintreten. Kann der Algorithmus von Ricart und Agrawala zu Deadlocks führen? Erklären Sie Ihre Antwort.

Aufgabe 4: Programmierung: Ricart/Agrawala

Im Archiv `u09Files.zip`[1] auf der Vorlesungswebseite finden Sie die Realisierung eines Prozeß-Systems mit mehreren Server-Prozessen. Die Server erhalten zu Beginn vom Client über eine Nachricht genau eine Aufgabe, die sie lokal bearbeiten. Zur Bearbeitung müssen sich die Server in einen kritischen Abschnitt begeben, in dem eine Datei (`ressource.txt`) geschrieben wird.

Realisieren Sie auf Basis des gegebenen Codes den Algortihmus zum wechselseitigen Ausschluss nach Ricart und Agrawala, so daß zu jeder Zeit immer nur ein Server in die Datei schreiben kann. Vervollständigen Sie dazu die Methoden `lockResource()`, `unlockResource()` und `handleMutexMessage()` in `Server.java` und ergänzen Sie ggf. notwendige Deklarationen und Initialisierungen. Zur Vereinfachung dürfen Sie annehmen, daß ein Server nur genau einmal einen Auftrag vom Client bekommt, während der Bearbeitung des Auftrags also kein neuer Auftrag eintreffen kann.

Die für den Algortihmus notwendige Realisierung von Lamport-Zeitstempeln ist bereits vollständig vorgegeben. Der Absender einer Nachricht kann ebenfalls ermittelt werden (Methode `getSender()` der Klasse `Message`).

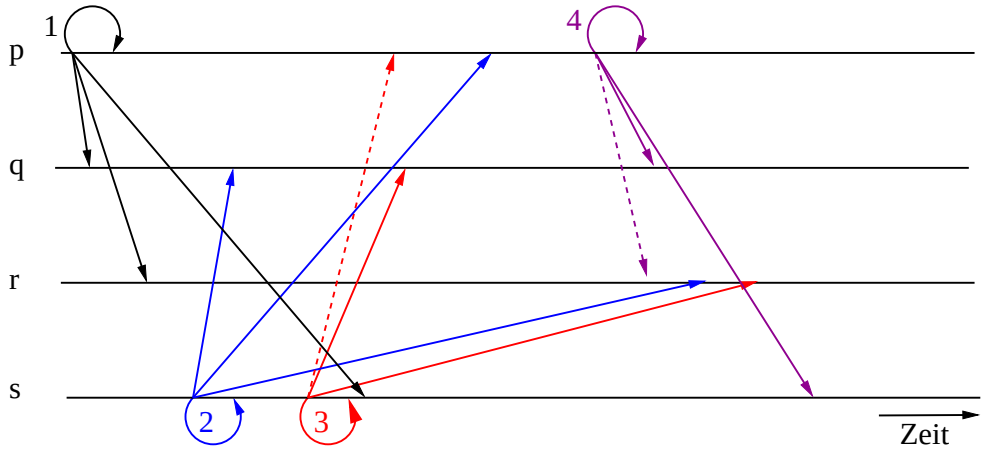
[1] `http://www.bs.informatik.uni-siegen.de/web/wismueller/vl/vs/u09Files.zip`

Testen Sie Ihre Lösung mit unterschiedlich vielen, gleichzeitigen Aufträgen und unterschiedlich vielen Servern. Die Zahl der Aufträge kann beim Client als Kommandozeilen-Argument angegeben werden (maximal soviele, wie Server vorhanden sind).

Aufgabe 5: Reihenfolgegarantien beim Multicast

In der Vorlesung wurden verschiedene Varianten für Reihenfolge-Garantien beim Multicast vorgestellet (s. Kap. 7.3).

a) In der Abbildung sind 4 Prozesse (p, q, r, s) gezeigt und vier Multicast-Nachrichten $(1, 2, 3, 4)$.



Nachricht 4 ist im Beispiel kausal abhängig von Nachricht 3.
Wann muß die Auslieferung der Nachrichten 3 und 4 stattfinden, so dass eine FIFO und eine kausale Sortierung für alle Nachrichten gewährleistet wird? Hinweis: Sie müssen nur die gestrichelten Pfeile richtig zeichnen. Ist dann auch eine vollständige Sortierung gewährleistet?

b) Um die Multicast-Sortierung zu verdeutlichen, betrachten wir die Anwendung „ elektronisches schwarzes Brett“, bei der die Benutzer Nachrichten auf einem schwarzen Brett hinterlassen. Jeder Benutzer führt einen Applikationsprozess für das schwarze Brett aus. Es gibt mehrere Diskussionsthemen, zu denen jeweils eine eigene Prozeßgruppe gehört. Der Prozess eines Benutzers ist Mitglied in der Gruppe für das Thema, an dem dieser interessiert ist, sodass der Benutzer nur Nachrichten zu dem jeweiligen Thema erhält. Wenn ein Benutzer eine Nachricht auf einem schwarzen Brett anbringt, multicastet die Applikation diese an die entsprechende Gruppe.

Ein Benutzer X könnte dann folgende Anzeige aus dem Programm für das schwarze Brett erhalten:

```
Schwarzes Brett zum Thema: Kommunikation in verteilten Systemen

Nachricht      von             Titel der Nachricht
-------------------------------------------------------------
   27          A. Bauer        UDP & TCP Kommunikation
   28          C. Fritz        RPC Prinzip
   29          G. Meier        Re: UDP & TCP Kommunikation
   30          K. Wim          RPC 1
   31          A. Bauer        Re: RPC Prinzip
   32          K. Wim          RPC 2
```

Beachten Sie, dass die Nachrichten, deren Themen mit *Re:* anfangen, nach den Nachrichten erscheinen, auf die sie sich beziehen.

Welche sortierte Multicast-Verfahren werden in der Applikation gebraucht und wo genau? Wenn der Multicast vollständig sortiert wäre, was kann man dann über die Nummerierung in der linken Spalte sagen?

Aufgabe 6: Kausal sortierter Multicast

Angenommen, ein System stellt zuverlässigen Multicast mit FIFO-Sortierung zur Verfügung. Man könnte auf die Idee kommen, darauf aufbauend einen kausal sortierten Multicast wie folgt zu implementieren:

- jede Nachricht wird mit dem Lamport-Zeitstempel des Sende-Ereignisses versehen,
- die Empfänger-Knoten liefern ankommende Nachrichten in einer Reihenfolge an die Empfängerprozesse aus, die konsistent mit der Ordnung der Zeitstempel ist.

a) Begründen Sie, warum dieser Multicast kausal sortiert ist.

b) Wo ist das Problem bei dieser Implementierung? (**Tip**: wann kann eine Nachricht an den Empfängerprozess ausgeliefert werden?) Wie könnte es gelöst werden?

c) Begründen Sie, warum dieser Multicast nicht vollständig sortiert ist. Wie könnte man eine vollständige Sortierung erreichen?