UNIVERSITÄT SIEGEN

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

Aufgabenblatt 9
Musterlösung

Vorlesung Verteilte Systeme
Wintersemester 2024/25

Aufgabe 1: Algorithmus von Ricart und Agrawala

Angenommen, ein Prozess verweigert die Berechtigung und stürzt dann ab. Der anfordernde Prozess glaubt, er sei noch am Leben, aber die Berechtigung trifft niemals ein. Ein Ausweg ist, den Anforderer zu veranlassen, dass er nicht nur blockiert, sondern dass er eine bestimmte Zeidauer lang *schläft*, wonach er alle Prozesse abfragt, die die Berechtigung verweigert haben, um zu prüfen, ob sie noch laufen.

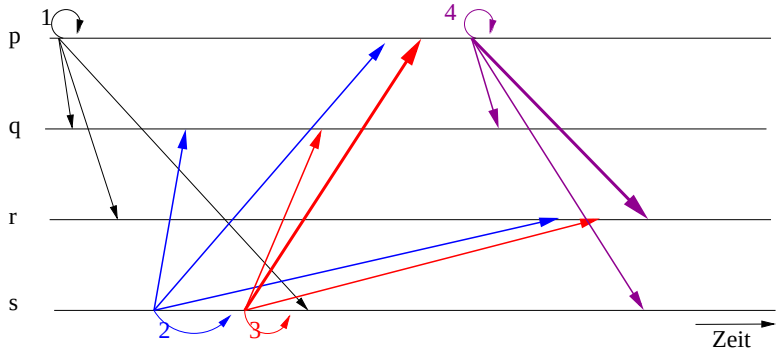
Aufgabe 2: Deadlocks - Ricart und Agrawala

Das ist von den Grundregeln abhängig. Wenn die Prozesse streng sequentiell in kritische Bereiche eintreten, d.h. ein Prozess in einem kritischen Bereich darf nicht versuchen, in einen weiteren kritischen Bereich einzutreten, gibt es keine Möglichkeit, dass er blockiert, während er eine Ressource (d.h. einen kritischen Bereich) hält, die ein anderer Prozess braucht. Das System ist Deadlock-frei. Kann andererseits Prozess 0 in den kritischen Bereich A eintreten und dann versuchen, in den kritischen Bereich B einzutreten, kann ein Deadlock auftreten, wenn ein anderer Prozess versucht, die kritischen Bereiche in umgekehrter Reihenfolge zu betreten. Der Algorithmus von Ricart und Agrawala selbst trägt nicht zu dem Deadlock bei, weil jeder kritische Bereich unabhängig von den anderen behandelt wird.

Aufgabe 3: Programmierung: Ricart/Agrawala (Pflichtaufgabe für 6 LP, Abgabe über moodle!)

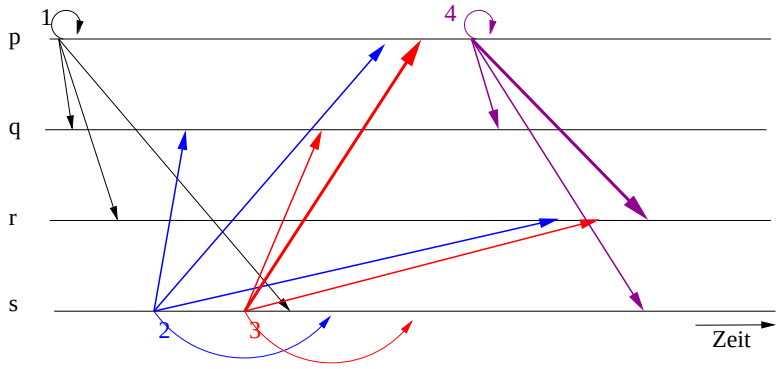
Aufgabe 4: Reihenfolgegarantien beim Multicast

a) Bei einem kausal sortierten Multicast muss die Nachricht 3 in allen Prozessen (insbes. Prozeß p) nach der Nachricht 2 eintreffen (da 3 kausal von 2 abhängt). Ebenso muß Nachricht 4 in allen Prozessen (insbes. Prozeß r) nach Nachricht 3 eintreffen.



Damit ist aber noch keine vollständige Sortierung gewährleistet (z.B. Prozeß p: Reihenfolge 1,2,3,4; Prozeß s: Reihenfolge 2,3,1,4).

Eine vollständige Sortierung wäre gegeben, wenn alle Nachrichten von allen Prozessen in derselben Reihenfolge ausgeliefert werden.



b) Damit der Benutzer alle Veröffentlichungen erhält, wird ein **zuverlässiger** Multicast benötigt.

Eine **FIFO-Reihenfolge** wird benötigt, damit die Veröffentlichungen eines Benutzer, z.B. A. Bauer, überall in derselben Reihenfolge empfangen werden. Die Benutzer können dann konsistent über die „zweite Veröffentlichung“ von A. Bauer diskutieren. Zudem könnte sich der zweite Beitrag ja auf den ersten beziehen.

Eine **kausale-Reihenfolge** wird ebenfalls benötigt, weil die Nachrichten, deren Themen mit *Re:* anfangen, nach den Nachrichten erscheinen sollten, auf die sie sich beziehen. Andernfalls könnte eine Verzögerung einer Nachricht dazu führen, dass z.B. die Nachricht *Re: RPC Prinzip* vor der ursprünglichen Nachricht *RPC Prinzip* erscheint.

Wäre die Multicast-Auslieferung vollständig sortiert, wäre die Nummerierung in der linken Spalte konsistent zwischen den Benutzern. Dies ist aber nicht zwingend notwendig.

In der Praxis implementiert das USENET-System, das ein solches schwarzes Brett nachbildet, weder eine kausale noch eine vollständige Reihenfolge. Der dazu notwendige Kommunikationsaufwand würde ihre Vorteile bei weitem überwiegen.

Aufgabe 5: Vollständig sortierter Multicast

Ein zweiter Ansatz ist, die Nachricht sofort per Multicast zu versenden, aber die Auslieferung zu verschieben, bis der Sequenzer eine Sequenznummer dafür gemulticastet hat. Das passiert, nachdem die Nachricht vom Sequenzer empfangen wurde.

Ein dritter Ansatz ist, zuerst eine Sequenznummer vom Sequenzer zu ermitteln, und die Nachricht dann zu multicasten.

Der erste Ansatz (die Operationen an den Sequenzer zu senden) beinhaltet das Senden einer Punkt-zu-Punkt-Nachricht mit der Operation sowie eine Multicast-Nachricht.

Für den zweiten Ansatz braucht man zwei Multicast-Nachrichten: eine mit der Operation und eine mit der Folgenummer.

Der dritte Ansatz kostet eine Punkt-zu-Punkt-Nachricht mit der Sequenznummer, gefolgt von einer Multicast-Nachricht, die die Operation enthält.

Aufgabe 6: Kausal sortierter Multicast (Pflichtaufgabe für 6 LP, Abgabe über moodle!)