Aufgabe 5: Algorithmen für gegenseitigen Ausschluß in verteilten Systemen

Literatur:

[1] Singhal, M.; Shivaratri, N. G.
Advanced Concepts in Operating Systems
McGraw-Hill, Inc: New York et al. 1994

a) Was bedeutet "logische Zeit"?

b) Welche Eigenschaften müssen bei allen Verfahren zum gegenseitigen Ausschluß untersucht werden?

c) Welche zwei Klassen werden unterschieden?

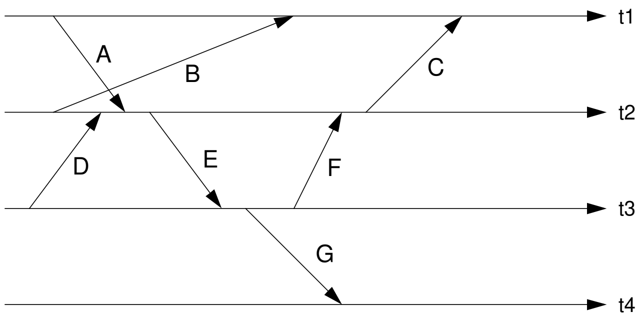
d) Beschreiben Sie kurz einige Algorithmen!

Lösungsvorschlag Aufgabe 5: Gegenseitiger Ausschluß in verteilten Systemen

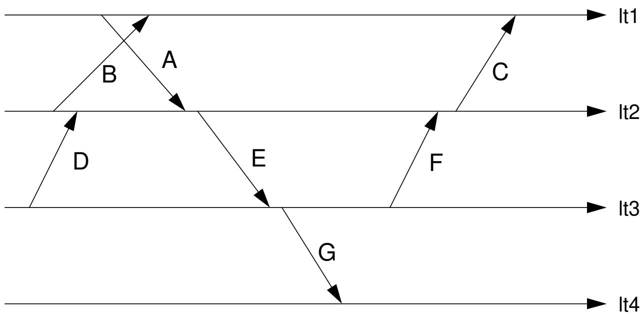
a) Was bedeutet "logische Zeit"?

Jedem Ereignis wird eine logische Zeit zugeordnet. Auf der logischen Zeit ist eine partielle Ordnung definiert. Es müssen folgende Eigenschaften erfüllt sein:

- Ein Ereignis A, das durch ein anderes Ereignis B bedingt ist, hat eine spätere logische Zeit. (Kausaler Zusammenhang)
- Ereignisse, die sich nicht kausal bedingen, können unvergleichbar sein; sie können aber auch durch entsprechende Definitionen in eine totale Ordnung gebracht werden.
- Sich bedingende Ereignisse entstehen z.B. durch Nachrichtenaustausch.



Eine mögliche Definition einer logischen Zeit mit totaler Ordnung wäre:
Ds, Bs, De, As, Be, Ae, Es, Ee, Gs, Ge Fs, Fe, Cs, Ce, also:



b) Welche Eigenschaften müssen bei allen Verfahren zum gegenseitigen Ausschluß untersucht werden?

Sicherheit (safety property):
Der gegenseitige Ausschluß wird nie verletzt.

Lebendigkeit (liveness property)
Jede Anforderung wird erfüllt.

Eine Folge der Sicherheitsbedingung ist, daß bei dem Ausfall des Knoten, der sich im kritischen Abschnitt befindet, das ganze System beendet werden muß, da keine Aussagen über den Zustand des kritischen Abschnitts mehr möglich sind. Muß hier eine Fehlerbehandlung durchgeführt werden, müssen höhere Konzepte wie z.B. *Transaktionen* verwendet werden. Dort kann das System auf einen definierten (Zwischen-)Zustand zurückgesetzt werden, von dem aus nach einer Rekonfiguration weitergemacht werden kann. Oft werden diese höheren Konzepte statt dem 'einfachen' gegenseitigen Ausschluß eingesetzt.

c) Welche zwei Klassen werden unterschieden?

Erlaubnisbasierte Verfahren (Permission based)

Die grundlegende Idee:
Wenn ein Prozeß den kritischen Abschnitt betreten will, fragt er andere Knoten um Erlaubnis, den Abschnitt zu betreten, und wartet, bis er die Erlaubnis bekommt.

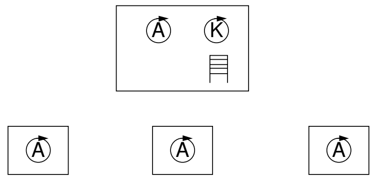
Tokenbasierte Verfahren (token based)

Die grundlegende Idee:
Das Recht, den kritischen Abschnitt zu betreten, wird übertragen auf ein spezielles Objekt, das im System eindeutig ist - das Token.
Es muß nur noch darauf geachtet werden, daß jeder Prozeß das Token erhält. Hierfür gibt es zwei Möglichkeiten: das Token ist aktiv und zirkuliert selbständig im Netz oder das Token selbst ist passiv und es muß explizit angefordert werden.
Es muß mit absoluter Sicherheit ausgeschlossen werden, daß das Token verdoppelt wird, da bei einer Verdoppelung kein gegenseitiger Ausschluß mehr gewährleistet ist.

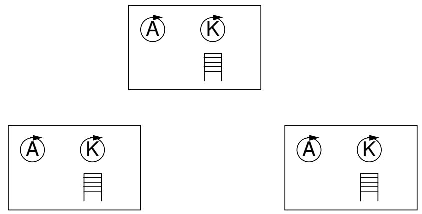
Wird die Schnittmenge aus beiden Klassen gebildet, so ergibt sich ein Algorithmus mit zentralem Koordinator.

d) Beschreiben Sie kurz einige Algorithmen!

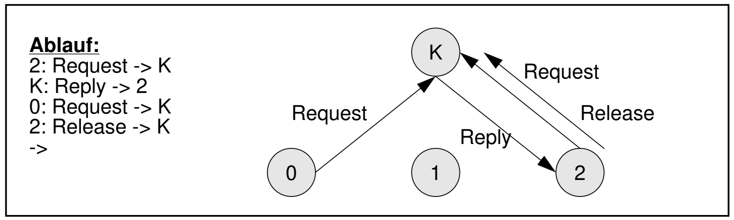
Software-Struktur bei Zentralem Koordinator:



Software-Struktur bei verteilten Algorithmen:



- <u>Zentralisiert</u>
  Ein einziger ausgezeichneter Prozeß vergibt die Berechtigung bzw. das Token, um den kritischen Abschnitt zu betreten. Jeder Prozeß muß sich an ihn und sonst keinen wenden.

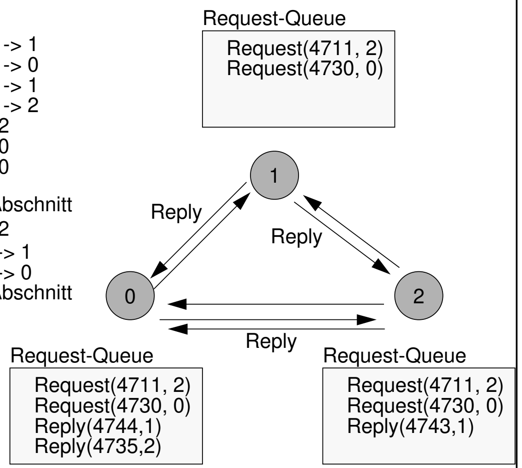


- Der zentralisierte Algorithmus ist ein Extremfall des generalisierten Algorithmus von Sanders, das andere Extrem ist der Algorithmus von Lamport.

- Lamport
  Voraussetzung ist, daß eine totale Ordnung der Ereignisse existiert, die nach dem Verfahren von Lamport für logische Uhren zusammen mit der Prozeßidentifikation erzielt werden kann. Alle Nachrichten tragen einen solchen Zeitstempel. Alle N Prozesse sind beteiligt.
  Wenn ein Prozeß den kritischen Abschnitt betreten will, schickt er eine *request resource* -Nachricht an alle anderen Prozesse. Diese fügen die Nachricht in eine nach Zeitstempeln geordnete Warteschlange ein und schicken eine Bestätigung mit Zeitstempel zurück. Ein Prozeß darf den kritischen Abschnitt betreten, wenn in seiner eigenen Warteschlange seine eigene *request resource* -Nachricht an erster Stelle steht und er von jedem anderen Prozeß eine Nachricht (*request* oder *reply*) mit einem neueren Zeitstempel erhalten hat.
  Durch *release resource* -Nachrichten werden die *request resource* -Nachrichten wieder entfernt und ein anderer Prozeß kann den kritischen Abschnitt betreten.

  Das Versenden von Nachrichten wird als zuverlässig angenommen. Es treten bei der Kommunikation zwischen zwei Knoten keine Überholvorgänge auf (FIFO-Eigenschaft), insbesondere darf ein Reply kein Request überholen.
  Bei Ausfall eines Rechners kann kein Prozeß den kritischen Abschnitt betreten.

Ablauf:
2: Request(4711, 2) -> 1
2: Request(4711, 2) -> 0
0: Request(4730, 0) -> 1
0: Request(4730, 0) -> 2
1: Reply(4743,1) -> 2
1: Reply(4744,1) -> 0
2: Reply(4735,2) -> 0
->
2: betritt kritischen Abschnitt
0: Reply(4737,0) -> 2
2: Release(4800,2) -> 1
2: Release(4801,2) -> 0
0: betritt kritischen Abschnitt



- Ricart & Agrawala
  Es gelten die gleichen Voraussetzungen wie beim Algorithmus von Lamport.
  Wenn ein Prozeß den kritischen Abschnitt betreten will, schickt er eine Nachricht an alle anderen Prozesse.
  Diese haben drei Möglichkeiten zu reagieren:
  Wenn sich der Empfänger nicht in einem kritischen Abschnitt befindet und ihn auch nicht betreten will, schickt er ein Reply zurück.
  Wenn er sich in seinem kritischen Abschnitt befindet, dann schickt er keine Antwort und fügt die Anforderung in seine Warteschlange ein.
  Wenn er seinen kritischen Abschnitt betreten will, es aber noch nicht getan hat, vergleicht er die Zeitstempel. Wenn seiner neuer ist, schickt er eine Antwort und läßt seine eigene Anforderung in der Warteschlange. Falls er den älteren Zeitstempel hat, fügt er die Anforderung in seine Warteschlange ein und sendet nichts.
  Ein Prozeß darf seinen kritischen Abschnitt betreten, wenn er die Bestätigung von allen anderen Prozessen hat.
  Beim Verlassen des KA werden verzögerte Replies zugestellt

  Das Versenden von Nachrichten wird als zuverlässig angenommen. Die FIFO-Eigenschaft wird nicht angenommen.
  Bei Ausfall eines Rechners kann kein Prozeß den kritischen Abschnitt betreten.

Ablauf:
2: Request(4711, 2) -> 1
2: Request(4711, 2) -> 0
0: Request(4730, 0) -> 1
0: Request(4730, 0) -> 2
1: Reply -> 2
1: Reply -> 0
0: Reply -> 2
->
2: betritt kritischen Abschnitt
2: Reply -> 0
0: betritt kritischen Abschnitt