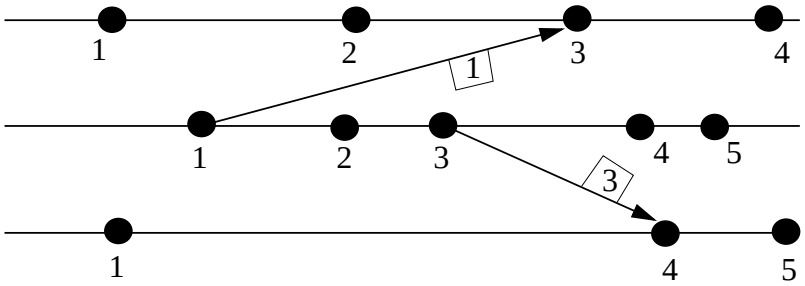
Logische Uhren von Lamport

Commun. ACM 1978:
Time, Clocks, and the Ordering of Events in a Distributed System

C: (E,<) --> (**N**,<) Zuordnung von Zeitstempeln

Kausal-relation

e < e' ==> C(e) < C(e') Uhrenbedingung



Protokoll zur Implementierung der Uhrenbedingung:

- Lokale Uhr (= Zähler) *tickt* "bei" *jedem* Ereignis
- Sendeereignis: Uhrwert mitsenden (*Zeitstempel*)
- Empfangsereignis: *max*(lokale Uhr, Zeitstempel)

zuerst! danach "ticken"

Behauptung:

Protokoll respektiert Uhrenbedingung

Beweis: Kausalitätspfade sind monoton...

Lamport-Zeit: Nicht-Injektivität

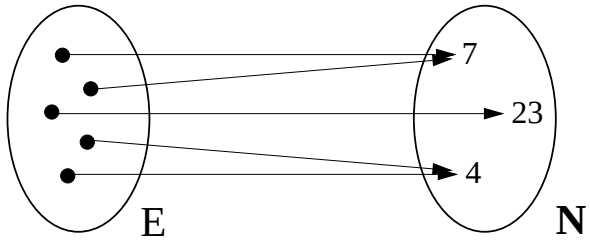


Abbildung ist nicht injektiv

- Wichtig z.B. für: "Wer die kleinste Zeit hat, gewinnt"

- Lösung:

  Lexikographische Ordnung (C(e),i), wobei i die Prozessnummer bezeichnet, auf dem e stattfindet

  Ist injektiv, da alle lokalen Ereignisse verschiedene Zeitstempel C(e) haben ("break ties")

  - *lin*. Ordnung $(a,b) < (a',b') \Leftrightarrow a<a' \vee a=a' \wedge b<b'$

  --> alle Ereignisse haben *verschiedene* Zeitstempel

  --> Kausalitätserhaltende Abb. (E,<) --> (**N**×**N, <)**

Jeder (nicht-leere) Menge von Ereignissen hat ein eindeutig "frühestes"!

Wechselseitiger Ausschluss

- Typischerweise exklusive Betriebsmittel
  - z.B. konkrete Betriebsmittel wie gemeinsamer Datenbus
  - oder abstrakte Betriebsmittel wie z.B. "Termin" in einem (verteilten) Terminkalendersystem
  - "kritischer Abschnitt" in einem (nebenläufigen) Programm

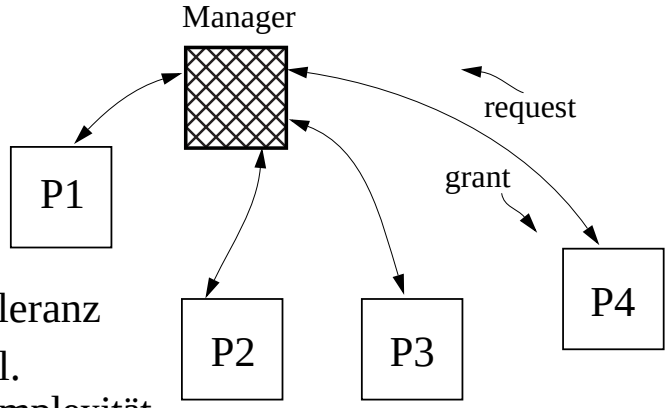
- Bei Einprozessormaschinen, shared memory etc.
  - Semaphore oder ähnliche Mechanismen

    ==> Betriebssystem-Theorie

    ==> interessiert uns hier nicht!

  - grundsätzlich interessant allerding: was sind die elementaren Basis-mechanismen, um wechselseitigen Ausschluss realisieren zu können?

- Nachrichtenbasierte Lösung, die uns nicht interessiert, da asymmetrisch ("zentralisiert"):

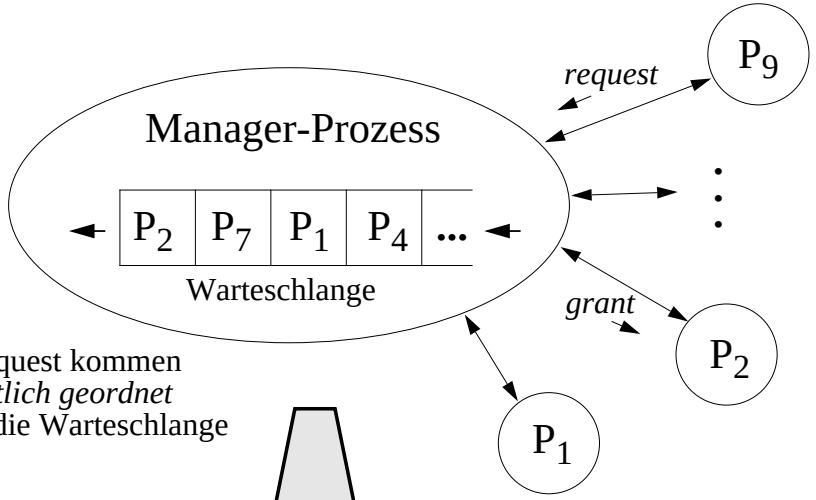


--> Engpass

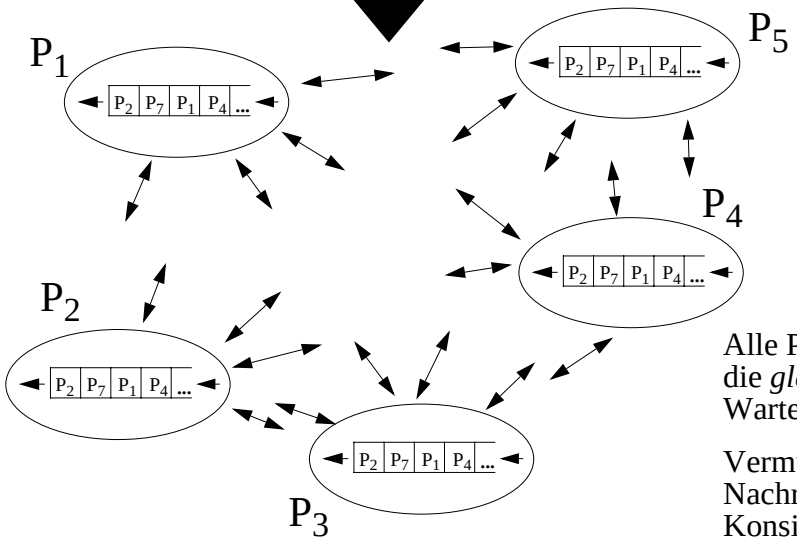
--> keine Fehlertoleranz

--> aber billig bzgl. Nachrichtenkomplexität

Replizierte Warteschlange?



Request kommen *zeitlich geordnet* in die Warteschlange



- Lässt sich mit "logischer Zeit" realisieren

Alle Prozesse sollen die *gleiche Sicht* der Warteschlange haben

Vermutlich viele Nachrichten, um Konsistenz zu gewähren; diese müssen ausserdem "geordnet" (=?) ankommen

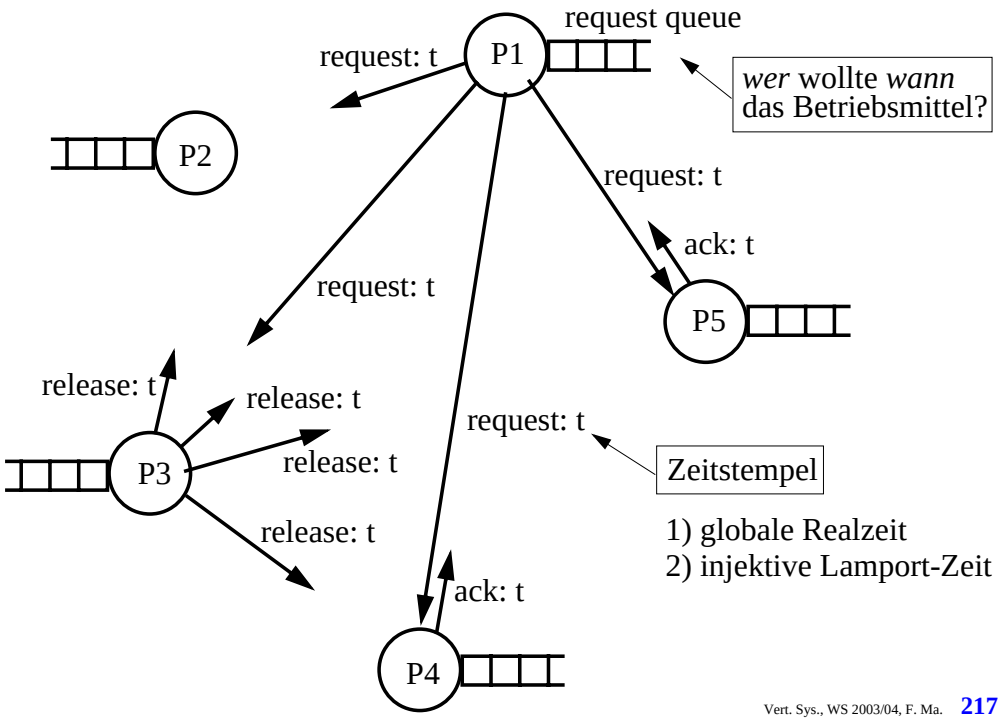
Anwendung logischer Zeit für den wechselseitigen Ausschluss

- Feste Anzahl von Prozessen
- Ein exklusives Betriebsmittel
- Synchronisierung mit request / release-*Nachrichten*
- Fairness: Jeder request wird schliesslich erfüllt

"request" / "release": --> vor Betreten / bei Verlassen des *kritischen Abschnittes*

Sehr schwache Fairnessanforderung

Idee: Replikation einer globalen request queue



1) globale Realzeit
2) injektive Lamport-Zeit

Der Algorithmus (Lamport '78):

- Voraussetzung: Verlustfreie FIFO-Kanäle
- *Alle* Nachrichten tragen Zeitstempel (eindeutige!)
- Request- und release-Nachrichten an *alle* senden ← broadcast

1) Bei "request" des Betriebsmittels: Mit Zeitstempel request in die eigene queue und an alle versenden.

2) Bei Empfang einer request-Nachricht: Request in eigene queue einfügen, ack versenden.
   - weiterer Nachr.typ
   - wieso notwendig?

3) Bei "release" des Betriebsmittels: aus eigener queue entfernen, release-Nachricht an alle versenden.

4) Bei Empfang einer release-Nachricht: Request aus eigener queue entfernen.

5) Ein Prozess darf das Betriebsmittel benutzen, wenn:
   - eigener request ist frühester in seiner queue und
   - hat bereits (irgendeine) spätere Nachricht von allen anderen Prozessen bekommen.

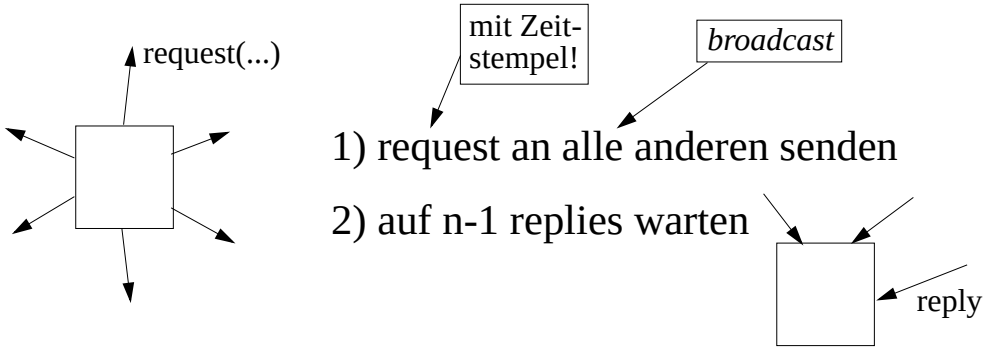
- Frühester request ist global eindeutig.
  ==> bei 5): sicher, dass kein früherer request mehr kommt (wieso?)

- 3 (n-1) Nachrichten pro "request"

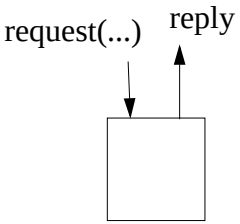
Denkübungen:
- was könnte man bei Nachrichtenverlust tun? (--> Fehlertoleranz)
- wo geht Uhrenbedingung / Kausaltreue der Lamport-Zeit ein?
- sind FIFO-Kanäle notwendig? (Szenario hierfür?)
- bei Broadcast: welche Semantik? (FIFO, kausal,...?)

Ricart / Agrawala 1981: "An Optimal Algorithm for..."

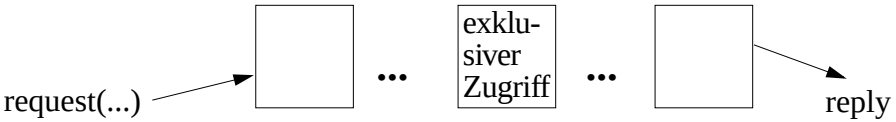
- 2(n-1) Nachrichten statt 3(n-1) beim Lamport-Verfahren:
  (*reply-Nachricht* übernimmt Rolle von release und ack)



- Bei Eintreffen einer request-Nachricht:



  - reply sofort schicken, wenn nicht beworben oder der Sender "ältere Rechte" (logische Zeit!) hat,
  - ansonsten reply erst später schicken, nach Erfüllen des eigenen requests ("verzögern")



- Ältester Bewerber setzt sich durch!

Denkübungen:

- Safety, liveness: Argumente?
- Wie oft muss ein Prozess maximal nachgeben? (--> Fairness)
- Sind FIFO-Kanäle notwendig?
- Geht es nicht mit weniger Nachrichten? ("optimal")