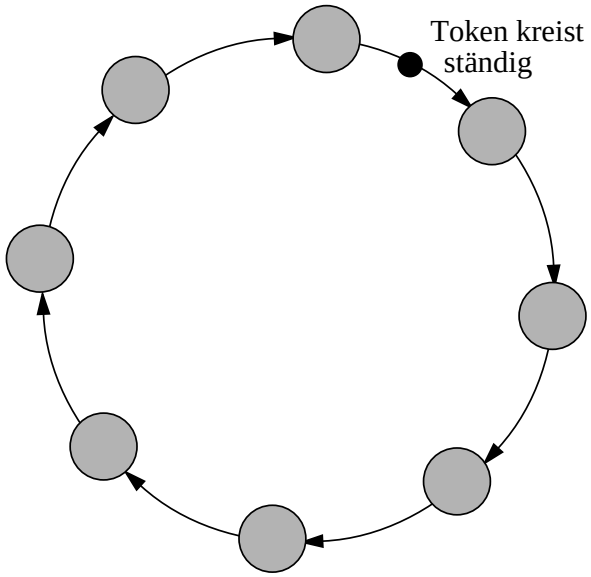
Die Token-Ring-Lösung



- Nur der Tokeninhaber darf
  - Safety ist klar
  - Liveness: Token muss weitergegeben werden
  - Fairness intuitiv gegeben

- Probleme?
  - Bei vielen Prozesse --> lange Wartezeiten, Gefahr von Tokenverlust
  - Anzahl der Einzelnachrichten nicht begrenzt (ständiges Kreisen)
  - Für jedes Betriebsmittel eigenes Token vorsehen

Token-Request-basierte Algorithmen

- Token soll nicht dauernd nutzlos unterwegs sein
  - Token wandert nur bei Bedarf

- Grundidee: "Token zu mir" an alle (?) anderen senden
  - per *broadcast* (falls entspr. Kommunikationsprimitiv existiert)
  - oder z.B. mittels *Echo-Algorithmus*
  - Aufwand ist hoch, wenn man nicht weiss, wo das Token sein könnte

- Fairness muss aber gewahrt bleiben

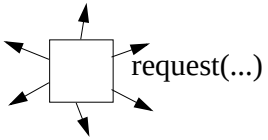
- Safety ist vergleichsweise trivial (Tokenbesitz)

- Liveness mit möglichst wenig Aufwand garantieren

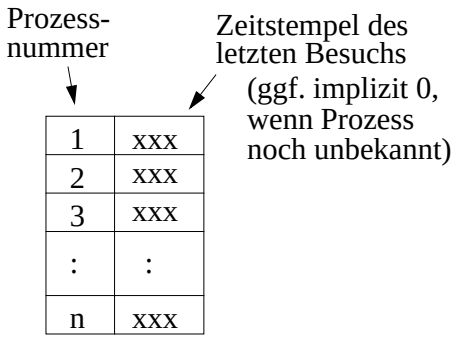
Ricart / Agrawala 1983

1) Es gibt ein einziges Token; nur Besitzer darf kritischen Abschnitt betreten

2) Request-Nachricht (mit Zeitstempel) an *alle* senden



3) Token hat "Gedächtnis":



4) Jeder Prozess registriert Anforderungen *aller* anderen

5) Nach Verlassen des kritischen Abschnitts wird das Token an denjenigen Prozess geschickt, der am "längsten" wartet (Anforderung muss jünger als Zeitstempel des letzten Besuchs beim Prozess sein)

--> Nachrichtenkomplexität: n (n-1 für request, + 1 Token)
(bzw. 0, wenn inzwischen kein anderer wollte)

Fragen:
- Wie lange muss ein Prozess max. (auf Mitbewerber) warten?
- Liveness? Fairness?
- Geht es auch mit weniger Nachrichten?
- Physische Zeit oder genügt z.B. auch Lamport-Zeit?