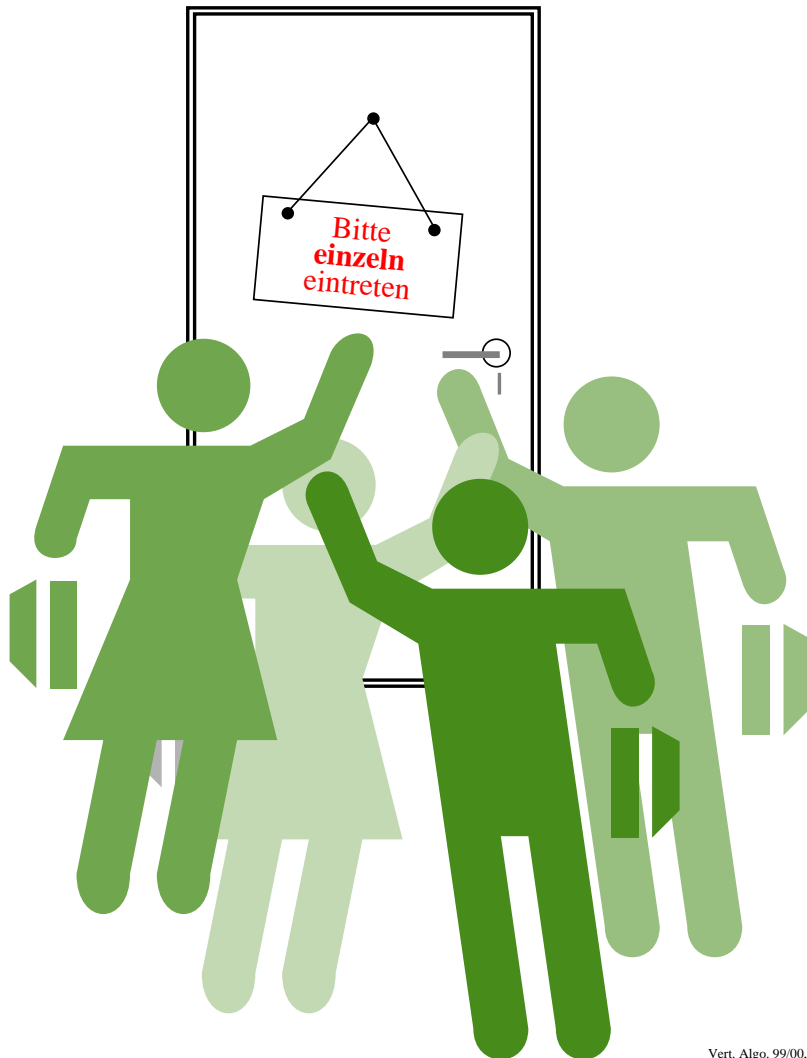


# Wechselseitiger Ausschluss

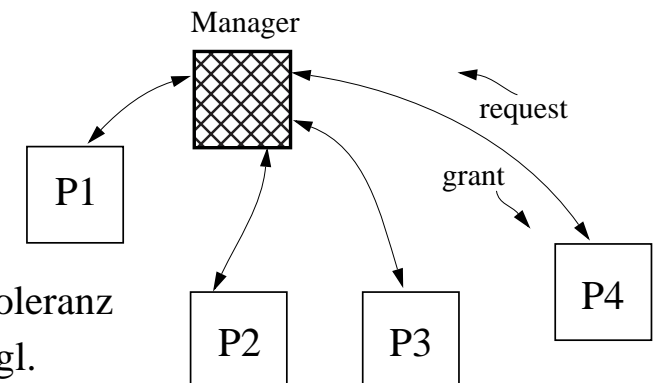


## Wechselseitiger Ausschluss

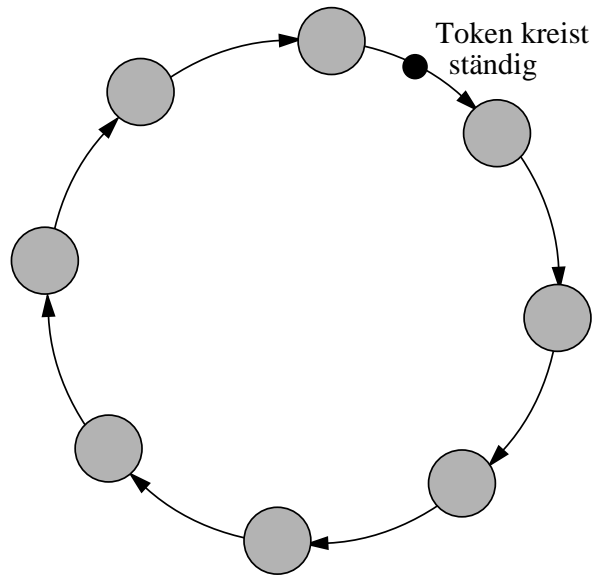
- Typischerweise exklusive Betriebsmittel
  - z.B. konkrete Betriebsmittel wie gemeinsamer Datenbus
  - oder abstrakte Betriebsmittel wie z.B. "Termin" in einem (verteilten) Terminkalendersystem
  - "kritischer Abschnitt" in einem (nebenläufigen) Programm
- Bei Einprozessormaschinen, shared memory etc.
  - Semaphore oder ähnliche Mechanismen
    - ==> Betriebssystem-Theorie
    - ==> interessiert uns hier nicht!
  - grundsätzlich interessant allerdings: was sind die elementaren Basismechanismen, um wechselseitigen Ausschluss realisieren zu können?

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

- > Engpass
- > keine Fehlertoleranz
- > aber billig bzgl. Nachrichtenkomplexität



# Die Token-Ring-Lösung



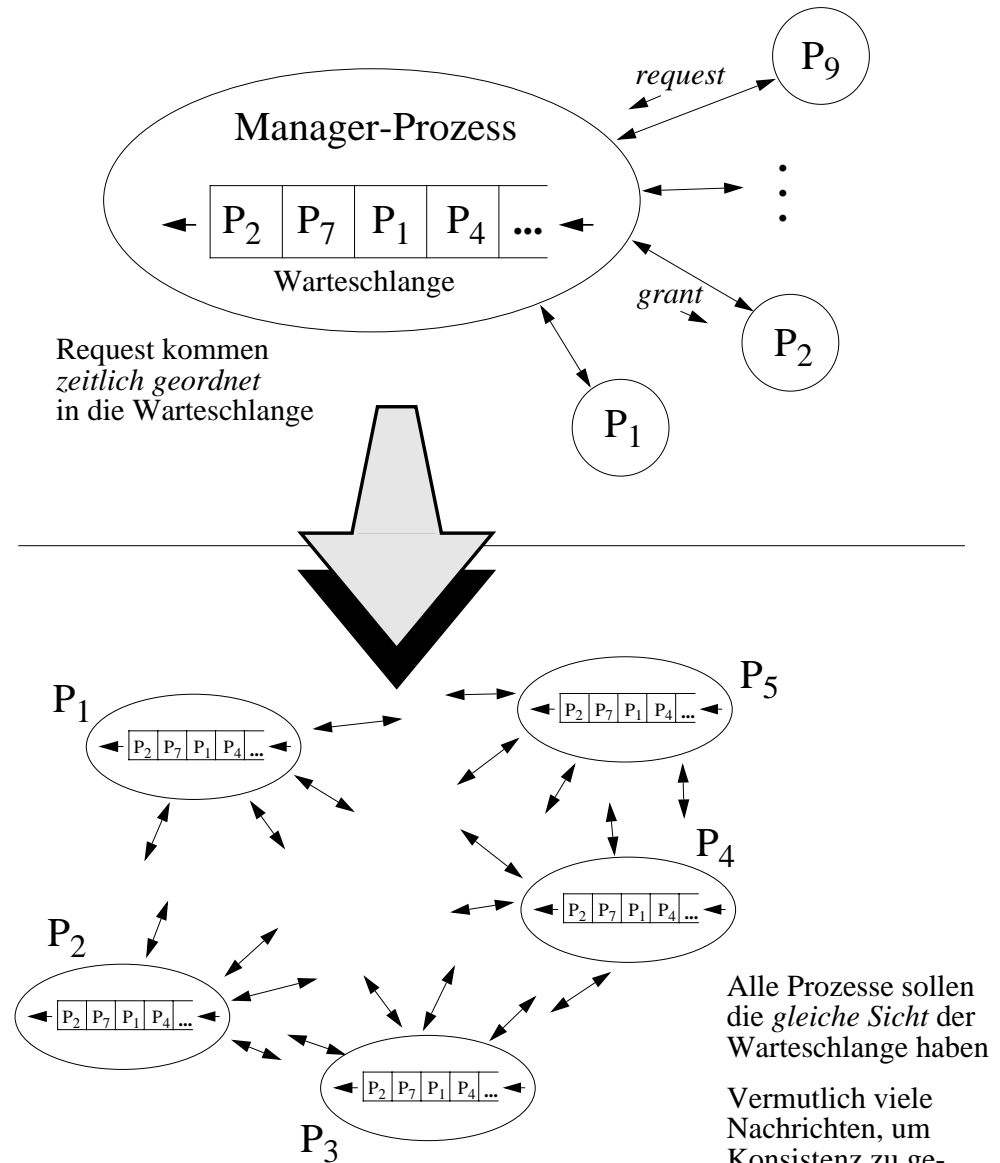
## - Nur der Tokeninhaber darf

- 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

# Replizierte Warteschlange?

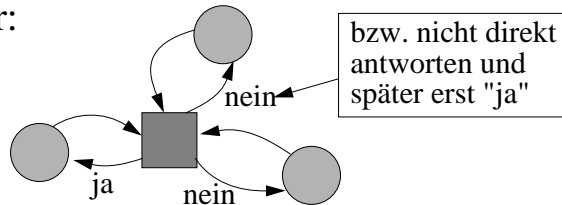


- Lässt sich mit "logischer Zeit" (--> später) realisieren
- > Algorithmus von Lamport (1978)

# Request-basierte Algorithmen

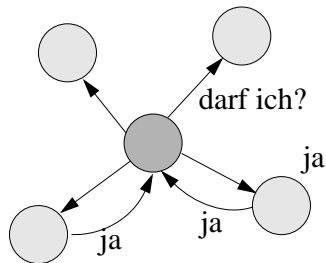
auch "permission-based" genannt

1) Zentraler Monitor:  
*Einen fragen*



2) *Alle fragen*

- dezentral und symmetrisch, aber viele Nachrichten



3) Kompromiss: dezentrale, symmetrische Lösung, bei der nur einige Prozesse um Erlaubnis gefragt werden?

- es müssen natürlich "ausreichend viele" sein

---

- Lassen sich diese Algorithmen als Ausprägung eines gemeinsamen allgemeinen Prinzips verstehen?

- > "ein für alle mal" verifizieren
- > abstraktere Sicht liefert hoffentlich tieferes Verständnis
- > ggf. neue Varianten mit interessanten Eigenschaften