Klassifikation: Token ⇔ Request

1) Token-basierte Lösungen

- Safety ist trivial
- Fairness bei Tokenweitergabe beachten
- Wie fordern Prozesse das Token an? ← --> unterschiedliche Lösungen
- Topologie
  - --> "Reiseweg des Tokens"
  - --> Zeitaufwand durch sequentielle Nachrichtenketten
- Fehlertoleranz:
  - wie Tokenverlust feststellen?
  - wer darf neues Token generieren? (Eindeutigkeit notwendig!)
- Nur anwendbar, wenn für das exklusive Betriebsmittel von vornherein ein Token eingerichtet wird --> nicht immer möglich!

- Beispiel für *a priori unbekannte exklusive Betriebsmittel:* "Reservierungszeiten für einen Tennisplatz"
  - "Ich benötige ihn übermorgen von 10.28 - 12.17 exklusiv"
  - Dafür lässt sich nicht von vornherein ein Token generieren!
  - Vielleicht: ein Token "Tennis <Zeitintervall>" dynamisch generieren?
  - Aber: wer garantiert, dass ein anderer dies nicht gleichzeitig tut?
  - Zurück zu einer zentralen Lösung mit allen Nachteilen? ("Tennisplatz als Monitor")
- Anderes Bsp. für ein "abstraktes Betriebsmittel": Terminvereinbarung mit einer beliebigen Menge von Teilnehmern
- Exklusives Generieren eines Tokens unter symmetrischen Bedingungen --> *Election-Problem* (--> später)

Klassifikation (2)

2) Request-basierte Lösungen

- wen sollen die Prozesse fragen? (request set)
- Safety sicherstellen!
- Deadlockfreiheit ist nicht trivial

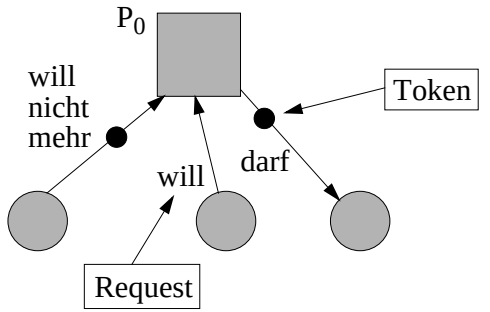
} auch hierfür verschiedene Lösungen

---

3) Zentraler Manager

bzw. Server oder Monitor

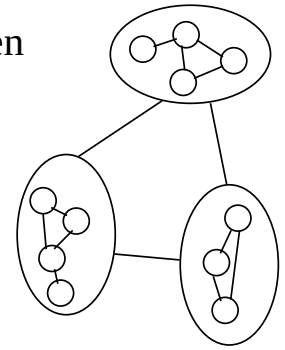
- ist entarteter Fall *beider* Klassen!



---

4) Hierarchische / hybride Verfahren

- bei grossen Systemen mehrstufig (mittels "Stellvertreter")
- auf verschiedenen Stufen / in verschiedenen Clustern ggf. unterschiedliche Verfahren



Vergleich der Nachrichtenkomplexität

(pro Betreten des kritischen Abschnittes bzw. Anforderung des Betriebsmittels)

LeLann (1977)	1 ... ∞
Lamport (1978)	3 (n-1)
Ricart / Agrawala (1981) ("an optimal algorithm...")	2 (n-1)
Ricart / Agrawala (1983)	n
Maekawa (1985)	$O(\sqrt{n})$
van de Snepscheut; Raymond; Naimi-Trehel (1987/89)	$O(\log_k n)$
zentraler Manager	2

- Offenbar soll es aber eben nicht alleine auf die Nachrichtenkomplexität ankommen!
- "Qualitative" Merkmale oft wichtiger!

Wechselseitiger Ausschluss: Kriterien

- Nachrichtenkomplexität

für die Qualität eines Lösungsalgorithmus

- Symmetrie
  - syntaktisch: gleicher Algorithmus für alle
  - semantisch: gleiche Last für alle etc.

Inwiefern würde man den O(log n)-Algorithmus als symmetrisch bezeichnen?

- Fehlertoleranz
  - Verhalten des Algorithmus bei Fehlern
    - z.B. Nachrichtenverlust
    - oder: nach Abbruch von aussen wegen Deadlock der Anwendung
  - Zusatzaufwand, um (etwa bei erkanntem Fehler) wieder einen konsistenten Zustand herzustellen
- Grad an Fairness
  - inwieweit wird zeitlich globale Reihenfolge der Requests eingehalten?
- Zeitbedarf zwischen Freigabe und Benutzung durch einen anderen Prozess
  - (minimale) Länge sequentieller Nachrichtenketten
  - Bsp.: bei O(log n)-Algorithmus ist Zeitbedarf auch O(log n) (statt O(1) wie bei einigen anderen Algorithmen)!
  - verschiedene Lastsituationen berücksichtigen:
    - schwache Last --> nur selten mehr als ein Konkurrent
    - hohe Last --> Betriebsmittel fast ständig in Benutzung
- Effizienz / Einfachheit der Implementierung
  - z.B.: wie wird broadcast / multicast ("request an alle") realisiert? (als effiziente Systemoperation; auf Ring; mit Echo-Algorithmus...)
  - wird eine spezielle Topologie vorausgesetzt (Ring, Baum,....) bzw. muss jeder Prozess jeden anderen kennen?