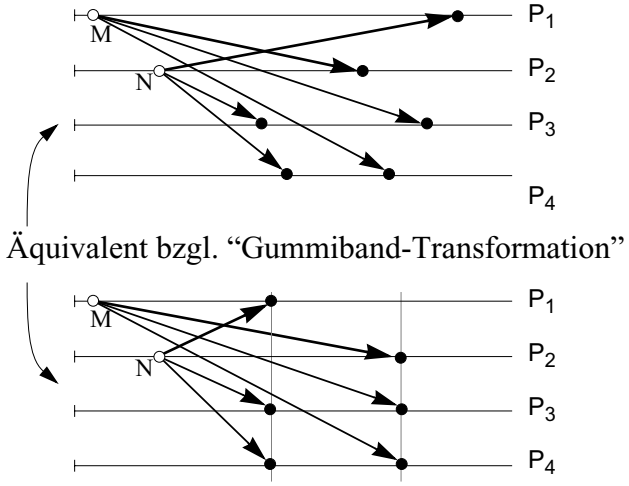
Atomarer bzw. “totaler” Broadcast

- *Totale Ordnung:* Wenn zwei Prozesse P_1 und P_2 beide die Nachrichten M und N empfangen, dann empfängt P_1 M vor N genau dann, wenn P_2 die Nachricht M vor N empfängt
- Beachte: Das Senden wird *nicht* als Empfang der Nachricht beim Sender selbst gewertet!
- Beachte: “Atomar” heisst hier *nicht* “alles oder nichts” (wie etwa beim Transaktionsbegriff von Datenbanken!)



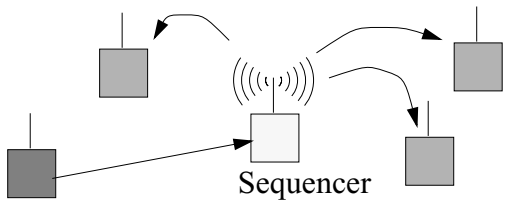
Anschaulich:

- Nachrichten eines Broadcasts werden “überall gleichzeitig” empfangen

Realisierung von atomarem Broadcast

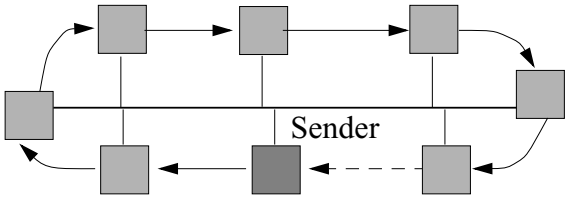
1) Zentraler „Sequencer“, der Reihenfolge festlegt

- ist allerdings ein potentieller Engpass!



- “Unicast” vom Sender zum Sequencer
- Multicast vom Sequencer an alle
- Sequencer wartet jew. auf alle Acknowledgements (oder genügt hierfür FIFO-Broadcast?)

2) Token, das auf einem (logischen) Ring kreist



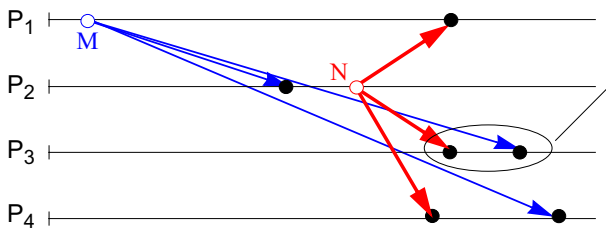
- Token = Senderecht (Token weitergeben!)
- Broadcast selbst könnte z.B. über ein zugrundeliegendes broadcastfähiges Medium erfolgen

- Token führt eine Sequenznummer (inkrementiert beim Senden), dadurch sind alle Broadcasts *global numeriert*
- Empfänger wissen, dass Nachrichten entsprechend der (in den Nachrichten mitgeführten Nummer) ausgeliefert werden müssen
- bei Lücken in den Nummern: dem Token einen Wiederholungswunsch mitgeben (Sender erhält damit implizit ein Acknowledgement)
- Tokenverlust (z.B. durch Prozessor-Crash) durch Timeouts feststellen (Vorsicht: Token dabei nicht versehentlich verdoppeln!)
- einen gecrashten Prozessor (der z.B. das Token nicht entgegennimmt) aus dem logischen Ring entfernen
- Variante (z.B. bei zu vielen Teilnehmern): Token auf Anforderung direkt zusenden (broadcast: “Token bitte zu mir”), dabei aber Fairness beachten (vgl. analoge Prinzipien bei Algorithmen für den wechselseitigen Ausschluss in Netzen → Vorlesung “Verteilte Algorithmen”)

- Geht es auch ohne zentrale Elemente (Sequencer, Token)?

Wie “gut” ist atomarer Broadcast?

1) Ist atomar auch kausal?

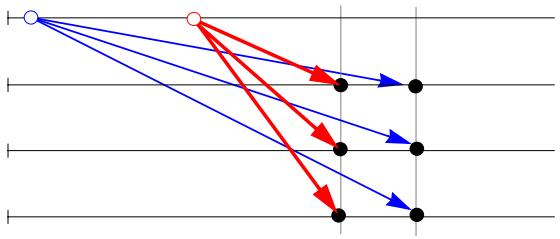


Nicht kausal!

Atomar: P3 und P4 empfangen beide M, N und zwar in gleicher Reihenfolge

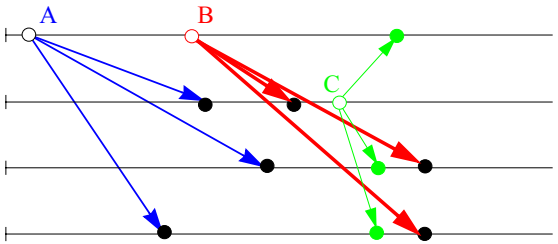
---

2) Ist atomar wenigstens FIFO?



---

3) Ist atomar + FIFO vielleicht kausal?



Bem.: 1) ist ebenfalls ein Gegenbeispiel, da M, N FIFO-Broadcast ist!

Kausaler atomarer Broadcast

- Fazit:
  - atomare Übermittlung $\not\Rightarrow$ kausale Reihenfolge
  - atomare Übermittlung $\not\Rightarrow$ FIFO-Reihenfolge
  - atomare Übermittlung + FIFO $\not\Rightarrow$ kausale Reihenfolge

- Vergleich mit speicherbasierter Kommunikation:
  - Kommunikation über gemeinsamen Speicher ist *atomar* (alle „sehen“ das Geschriebene gleichzeitig)
  - Kommunikation über gemeinsamen Speicher *wahrt Kausalität* (die Wirkung tritt unmittelbar mit der Ursache, dem Schreibereignis, ein)

- Vergleichbares Kommunikationsmodell per Nachrichten: *Kausaler atomarer Broadcast*
  - kausaler Broadcast + totale Ordnung
  - man nennt daher kausale, atomare Übermittlung auch *virtuell synchrone Kommunikation*
  - Denkübung: realisieren die beiden Implementierungen “zentraler Sequencer” bzw. “Token auf Ring” die virtuell synchrone Kommunikation?

Stichwort: Virtuelle Synchronität

- Idee: Ereignisse finden zu verschiedenen Realzeit-punkten statt, aber zur *gleichen logischen Zeit*
  - in Bezug auf die bis dahin empfangenen Nachrichten
  - vorläufig: „logische Zeit“ = Menge aller vergangenen Ereignisse

aber in welchem Sinne?

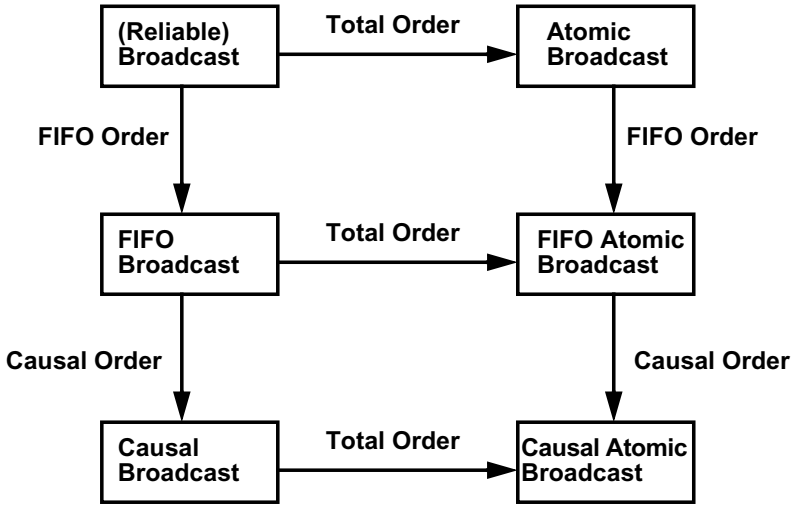
- *Innerhalb* des Systems ist synchron (im Sinne von “gleichzeitig”) und virtuell synchron *nicht unterscheidbar*
  - identische totale Ordnung aller Ereignisse
  - identische Kausalbeziehungen

- Folge: Nur mit Hilfe einer globalen Uhr könnte ein externer Beobachter den Unterschied feststellen

---

Den Begriff “logische Zeit” werden wir später noch genauer fassen
(mehr dazu dann wieder in der Vorlesung “Verteilte Algorithmen”)

Broadcast - schematische Übersicht



- Warum nicht ein einziger Broadcast, der alles kann? “Stärkere Semantik“ hat auch Nachteile:
  - Performance-Einbussen
  - Verringerung der potentiellen Parallelität
  - aufwendiger zu implementieren

- Bekannte “Strategie”:
  - man begnügt sich daher, falls es der Anwendungsfall gestattet, oft mit einer billigeren aber weniger perfekten Lösung
  - Motto: so billig wie möglich, so „perfekt“ wie nötig
  - man sollte aber die Schwächen einer Billiglösung kennen!

$\Rightarrow$ grössere Vielfalt $\Rightarrow$ komplexer bzgl. Verständnis und Anwendung

Multicast

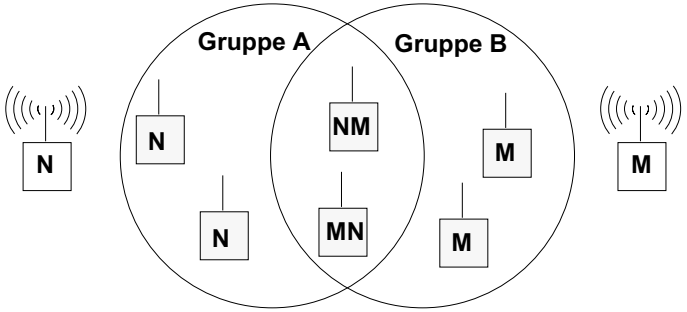
- Definition von Multicast (informell): "*Multicast ist ein Broadcast an eine Teilmenge von Prozessen*"
- Daher: Alles, was bisher über Broadcast gesagt wurde, gilt (innerhalb der Teilmenge) auch weiterhin:
  - zuverlässiger Multicast
  - FIFO-Multicast
  - kausaler Multicast
  - atomarer Multicast
  - kausaler atomarer Multicast
- Unterschied: Wo bisher "alle Prozesse" gesagt wurde, gilt nun "alle Prozesse innerhalb der Teilmenge"
- Wesentlich: Man kann *verschiedene* Teilmengen bilden
- Neu: Begriff der *Multicast-Gruppe* = Teilmenge von Prozessen

Multicast-Gruppen

- Zweck einer Multicast-Gruppe
  - "Selektiver Broadcast"
  - Vereinfachung der Adressierung (z.B. statt Liste von Einzeladressen)
  - Verbergen der Gruppenzusammensetzung (vgl. Mailbox/Port-Konzept)
  - "Logischer Unicast": Gruppen ersetzen Individuen (z.B. für transparente Replikation)
- Gruppenadressierung
  - *Explizite Benennung:* Sender nennt den Namen der Gruppe *("...grüsse den Kuckuckszuchtverein in Gimbelhausen")*
  - *Aufzählung der Mitglieder*: evtl. Multicast über Broadcast-Medium; gestattet dynamische Gruppen *("...grüsse Susi, Hugo & Erni")*
  - *Prädikatadressierung*: Ein potentieller Empfänger akzeptiert die Nachricht nur, wenn ein mitgesendetes Prädikat im lokalen Zustand des Empfängers 'wahr' ergibt *("...grüsse alle, die mich lieben")*
- Offene / geschlossene Gruppen
  - *Offen*: Nicht-Gruppenmitglieder dürfen Multicast-Nachrichten an die Gruppe senden
  - *Geschlossen*: Nur Gruppenmitglieder dürfen...
- Statische / dynamische Gruppen
  - *Dynamisch*: Gruppenzusammensetzung ändert sich ggf. im Laufe der Zeit: Gruppeneintritt, Gruppenaustritt, Ausfall eines Gruppenmitglieds; sind schwieriger zu verwalten als statische Gruppen

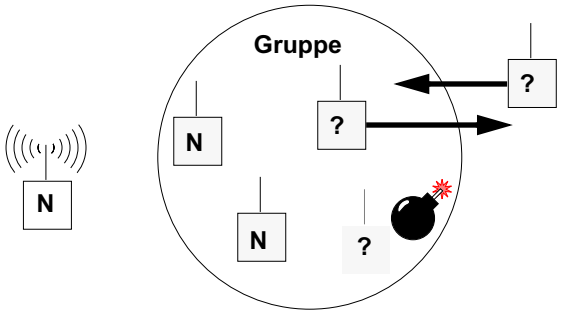
Grundprobleme bei Gruppen

- Gruppenüberlappung



- was geschieht (bzgl. Reihenfolge der Nachrichtenauslieferung etc.) genau im Überlappungsbereich?

- Gruppen-Management und Membership-Problem



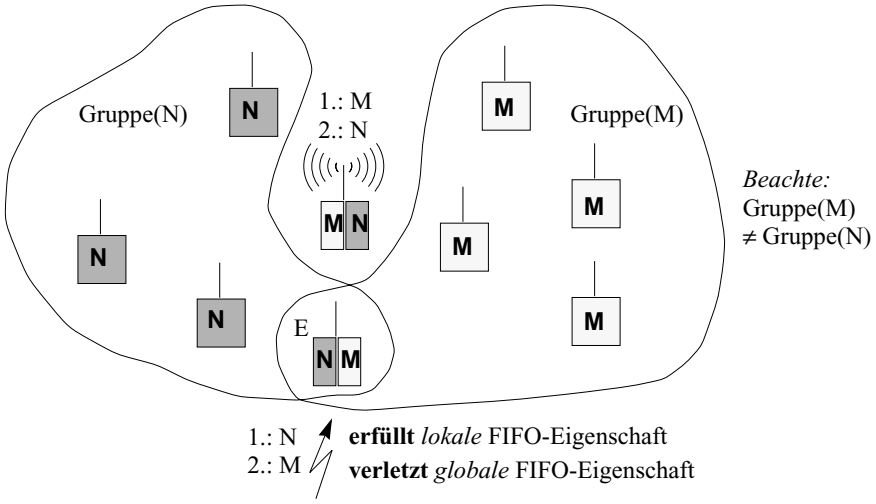
- dynamische Gruppe: wie sieht die Gruppe "momentan" aus?
- haben alle Mitglieder (gleichzeitig?) die gleiche Sicht?
- was tun bei Crashes?

Probleme der Gruppenüberlappung am Beispiel von FIFO-Multicast

was ist *sinnvoll?*

- Auf was genau soll sich die FIFO-Eigenschaft beziehen?
- Bezeichne *Gruppe(X)* die Multicast-Gruppe, an die die Nachricht X gesendet wird

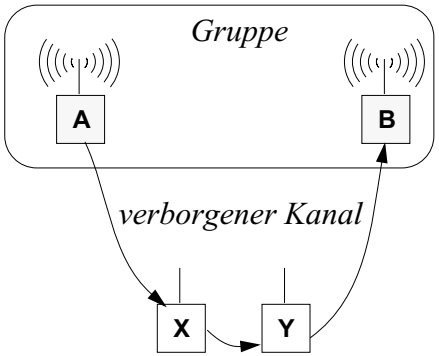
  - *Globale FIFO-Reihenfolge:* Wenn ein Prozess erst M und dann N sendet und ein Empfänger in Gruppe(M) die Nachricht N empfängt, dann muss er zuvor auch M empfangen haben
  - *Lokale FIFO-Reihenfolge:* Wenn ein Prozess erst M und dann N sendet mit Gruppe(N) = Gruppe(M) und ein Empfänger die Nachricht N empfängt, dann muss er zuvor auch M empfangen haben



- Analoge Unterscheidungen bzgl. lokaler / globaler Gültigkeit auch bei *kausalen* und *atomaren* Multicasts

Problem der “Hidden Channels”

- Kausalitätsbezüge verlassen (z.B. durch Gruppenüberlappung) die Multicast-Gruppe und kehren später wieder



?

- Soll nun das Senden von B als kausal abhängig vom Senden von A gelten?
- *Global* gesehen ist das der Fall, *innerhalb* der Gruppe ist eine solche Abhängigkeit jedoch nicht erkennbar
- Wie lautet die *sinnvolle* Definition von kausalem Multicast?

Übungsbeispiel zu Multicast (1)

(Nach einer Idee von Reinhard Schwarz)

Sinnvolle Definition von atomarem Multicast?

- **Lokale Totale Ordnung:** Wenn zwei Prozesse P_1 und P_2 beide die Nachrichten M und N empfangen mit Gruppe(M) = Gruppe(N), dann empfängt P_1 M vor N genau dann, wenn P_2 die Nachricht M vor N empfängt.
- **Paarweise Totale Ordnung:** Wenn zwei Prozesse P_1 und P_2 beide die Nachrichten M und N empfangen, dann empfängt P_1 M vor N genau dann, wenn P_2 die Nachricht M vor N empfängt.

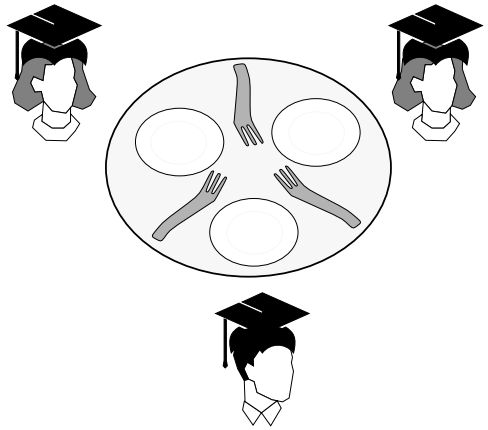
Fragen

- Wo braucht man solche Eigenschaften?
- Wann reichen die geforderten Eigenschaften, wann nicht?

Übungsbeispiel zu Multicast (2)

Beispiel: Problem der „speisenden Philosophen“

- Ein Philosoph **denkt** oder **speist**
- Zum Speisen benötigt ein Philosoph **rechte und linke Gabel.**
- Beim Denken gibt ein Philosoph beide Gabeln frei.



Wie stellt man sicher, dass die Philosophen nicht verhungern?

Übungsbeispiel zu Multicast (3)

Lösungsansatz:

Koordination der Gabelbenutzung per **paarweise atomarem Multicast**

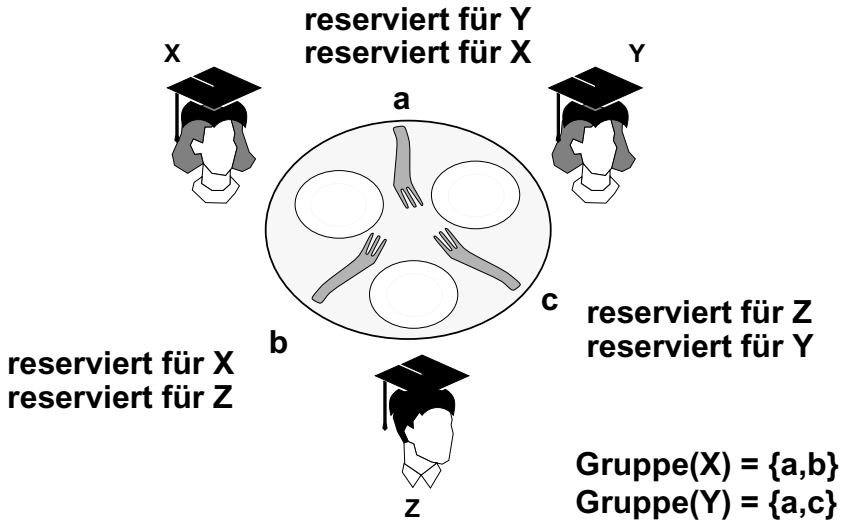
- Zum Essen sendet Philosoph X an seine benachbart liegenden Gabeln einen atomaren Multicast: *“reserviert für Philosoph X”*
- Gabel reserviert sich in der Reihenfolge der Anfragen.

→ **Durch paarweise totale Ordnung:**

- Reservierungen werden verklemmungsfrei vorgenommen, **oder?…**

Übungsbeispiel zu Multicast (4)

Typisches Szenario:



→ ***Reservierungen paarweise atomar:***
Kein Paar von Nachrichten von mehr als einer Gabel empfangen

→ ***Dennoch:*** Z wartet auf X, Z wartet auf Y, X wartet auf Z …
Deadlock!

Übungsbeispiel zu Multicast (5)

Atomarer Multicast – zweiter Versuch

wie bisher:

- Lokale Totale Ordnung: Wenn zwei Prozesse P_1 und P_2 beide die Nachrichten M und N empfangen mit Gruppe(M) = Gruppe(N), dann empfängt P_1 M vor N genau dann, wenn P_2 die Nachricht M vor N empfängt.
- Paarweise Totale Ordnung: Wenn zwei Prozesse P_1 und P_2 beide die Nachrichten M und N empfangen, dann empfängt P_1 M vor N genau dann, wenn P_2 die Nachricht M vor N empfängt.

neu:

- **Globale Totale Ordnung:** Paarweise Totale Ordnung +
Wenn eine Nachricht *M* von einem Prozess P_1 vor Nachricht *N* ausgeliefert wurde und *N* von einem Prozess P_2 vor Nachricht *O*, so liefert kein Prozess die Nachricht *O* vor *M* aus (Transitivität der Ordnungsrelation).

Membership

- Was bedeutet “alle Gruppenmitglieder”?
  - *Beitritt* (“join”) zu einer Gruppe während einer Übermittlung?
  - *Austritt* (“leave”) aus einer Gruppe während einer Übermittlung?
  - *Wechsel “korrekt” nach “fehlerhaft”* während einer Übermittlung?
- Beachte:
  - “Zufälligkeiten” (z.B. Beitrittszeitpunkt kurz vor / nach dem Empfang einer Einzelnachricht) sollen (soweit möglich) vermieden werden (→ Nichtdeterminismus; Nicht-Reproduzierbarkeit)
- Folge:
  - Zu jedem Zeitpunkt muss *Übereinstimmung* über *Gruppenzusammensetzung* und *Zustand* (“korrekt”, “ausgefallen” etc.) aller Mitglieder erzielt werden
- Frage:
  - Wie erzielt man diese Übereinkunft?

Wechsel der Gruppenmitgliedschaft

- Forderungen:

“...während...” gibt es nicht (→ “virtuell synchron”)

  - Eintritt und Austritt sollen *global atomar* erfolgen:
    - Die Gruppe muss bei allen (potentiellen) Sendern an die Gruppe hinsichtlich der Ein- und Austrittszeitpunkte jedes Gruppenmitglieds übereinstimmen
  - *globale Kausalität* soll gewahrt bleiben
- Realisierungsmöglichkeit:
  - konzeptuell führt jeder Prozess eine Liste mit den Namen aller Gruppenmitglieder
    - Realisierung als zentrale Liste (Fehlertoleranz und Performance?)
    - oder Realisierung als verteilte, replizierte Liste
  - massgeblich ist die zum Sendezeitpunkt gültige Mitgliederliste
  - Listenänderungen werden (virtuell) synchron durchgeführt:
    - bei zentraler Liste kein Problem
    - bei replizierten Listen: verwende *global kausalen, global atomaren Multicast*

      Schwierigkeit: *Bootstrapping-Problem* (mögliche Lösung: Service-Multicast zur dezentralen Mitgliedslistenverwaltung löst dies für sich selbst über einen zentralen Server)

Behandlung von Prozessausfällen

- Forderungen:
  - *Ausfall* eines Prozesses soll *global atomar* erfolgen:
    - Übereinstimmung über Ausfallzeitpunkt jedes Gruppenmitglieds
  - *Reintegration* nach einem vorübergehenden Ausfall soll *global atomar* erfolgen:
    - Übereinstimmung über Reintegrationszeitpunkt
  - *Globale Kausalität* soll gewahrt bleiben

- Realisierungsmöglichkeit:
  - Ausfallzeitpunkt:
    - Prozesse dürfen nur Fail-Stop-Verhalten zeigen: *"Einmal tot, immer tot"*
    - Gruppenmitglieder erklären Opfer per kausalem, atomarem Multicast übereinstimmend für tot: *"Ich sage tot – alle sagen tot!"*
    - Beachte: *"Lebendiges Begraben"* ist nicht ausschliessbar! (Irrtum eines "failure suspectors" aufgrund langsamer Nachrichten)
    - Fälschlich für tot erklärte Prozesse sollten unverzüglich Selbstmord begehen
  - Reintegration:
    - Jeder tote (bzw. für tot erklärte) Prozess kann der Gruppe nur nach dem offiziellen Verfahren ("Neuaufnahme") wieder beitreten

- Damit erfolgen Wechsel der Gruppenmitgliedschaft und Crashes in "geordneter Weise" für *alle* Teilnehmer

Multicast: Fazit

1. Wesentliche Neuerung gegenüber Broadcast:
   - Multicast-Gruppe
2. Gruppenüberlappungen schaffen Probleme:
   - z.B. lokale oder globale Gültigkeit von Ordnungsbeziehungen?
3. Änderung der Gruppenmitgliedschaft ist kritisch:
   - Lösen des Membership-Problems
   - Lösen des Bootstrap-Problems
4. Das Tolerieren von Prozessausfällen ist schwierig:
   - erfordert geeignetes Fehlermodell (z.B. Fail-Stop)
   - fälschliches Toterklären nicht immer vermeidbar