Verteilte Systeme

Synchronisation II

Prof. Dr. Oliver Haase

Überblick

Synchronisation 1

- Zeit in verteilten Systemen
- Verfahren zum gegenseitigen Ausschluss

Synchronisation 2

- Globale Zustände
- Wahlalgorithmen

globale Zustände

Überblick

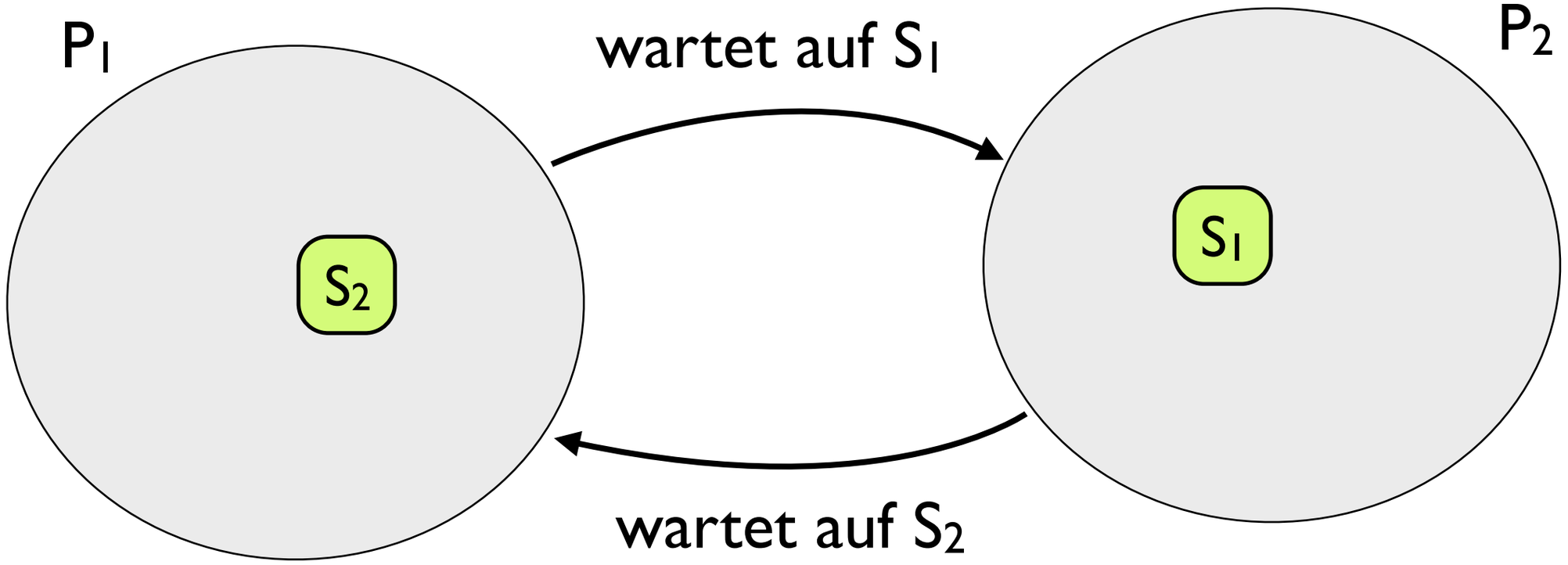
- Globale Zustände und deren Anwendung
- Verteilter Schnappschuss
- Der Begriff des Schnitts
- Der Algorithmus von Lamport und Chandy
- Beispiel
- Zusammenfassung

Globale Systemzustände

- Manche Anwendungen müssen über den Gesamtzustand des verteilten Systems Bescheid wissen

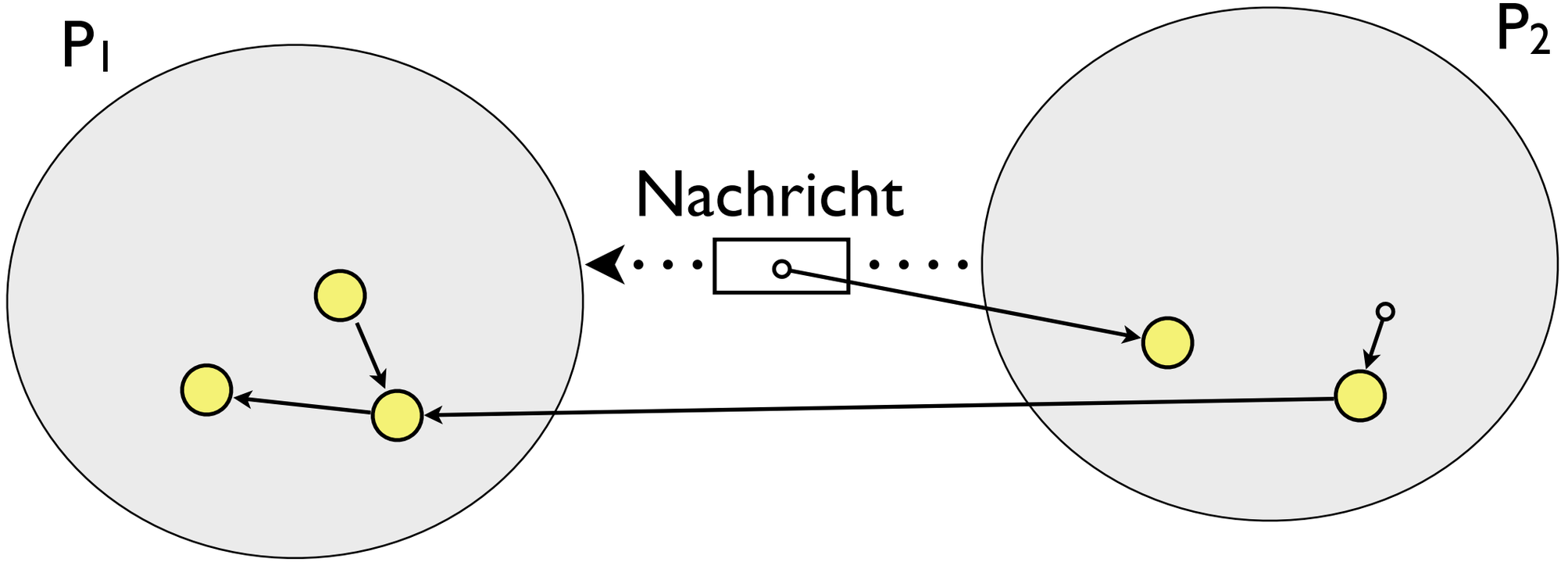
Anwendungsbeispiel 1

Deadlock Erkennung



Anwendungsbeispiel 2

verteilte Speicherbereinigung



Welche Objekte können speicherbereinigt werden?

Globale Systemzustände

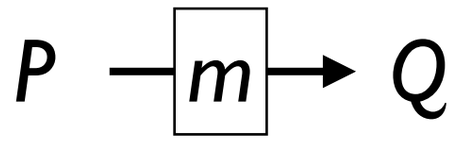
- Gesamtzustand des Systems besteht aus
  - lokalen Zuständen der Einzelkomponenten (deren Prozesse) und
  - allen Nachrichten, die sich zur Zeit in der Übertragung befinden.
- Diesen Zustand exakt zur selben Zeit zu bestimmen ist so unmöglich wie die exakte Uhrensynchronisation → es lässt sich kein globaler Zeitpunkt festlegen, an dem alle Prozesse ihre Zustände festhalten sollen *(jeder Prozess hat seine eigene Uhr*)

Verteilter Schnappschuss

- Wie kann man nun den globalen Zustand eines verteilten Systems ermitteln?
- Lösung von Chandy und Lamport (1985):
  - nimm lokale Snapshots in Einzelkomponenten zu (potentiell) unterschiedlichen Zeiten, aber so dass
  - alle zusammen konsistent sind.
- ergibt zusammen verteilten Schnappschuss (verteilte Momentaufnahme, engl. distributed snapshot)

Verteilter Schnappschuss

- Was bedeutet *zusammen konsistent*?
- Wie bei logischer Zeit: Synchronisation bzw. Konsistenz immer erforderlich, wenn Komponenten miteinander kommunizieren.
- Notation: *P* sendet Nachricht *m* an *Q*



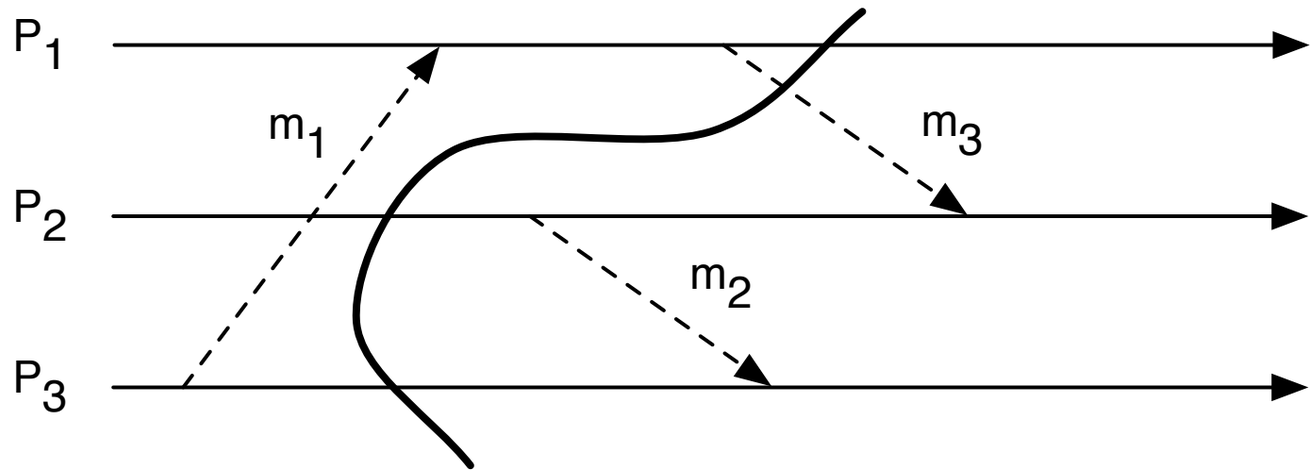
- Konsistenz bedeutet: Für alle *P*, *Q*, *m*:
  receive(m) ∈ Snapshot(Q) ⇒ send(m) ∈ Snapshot(P)

Verteilter Schnappschuss

- Anders ausgedrückt:
  Die Einzelkomponenten dürfen ihre lokalen Schnappschüsse zu unterschiedlichen Zeitpunkten machen, solange die Happens-Before-Relationship nicht verletzt ist.

Konsistente Schnitte

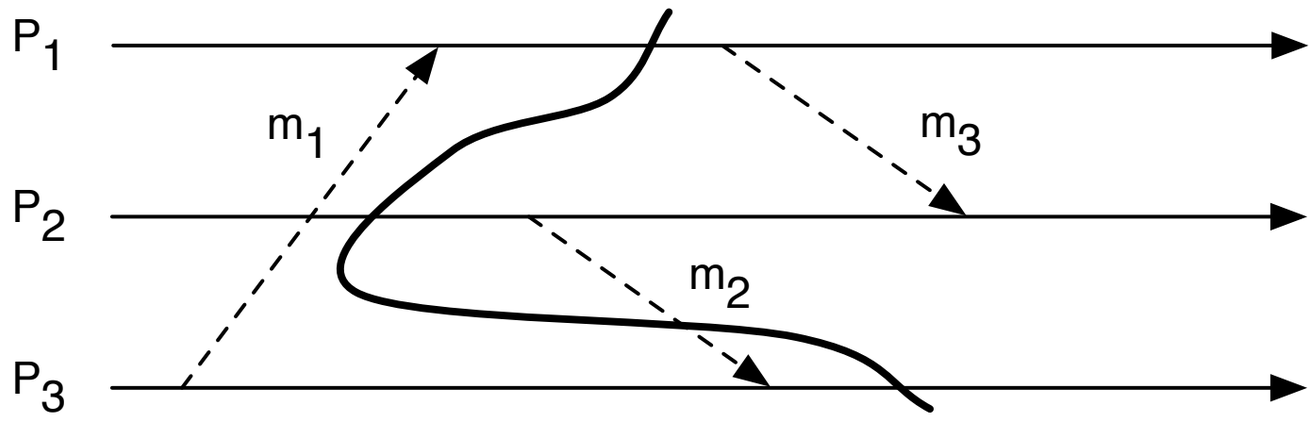
- Ein Schnitt gibt für jeden Prozess das letzte aufgezeichnete Ereignis an.
- Beispiele:



receive(m_1) ∈ Snapshot(P_1)
send(m_3) ∈ Snapshot(P_1)

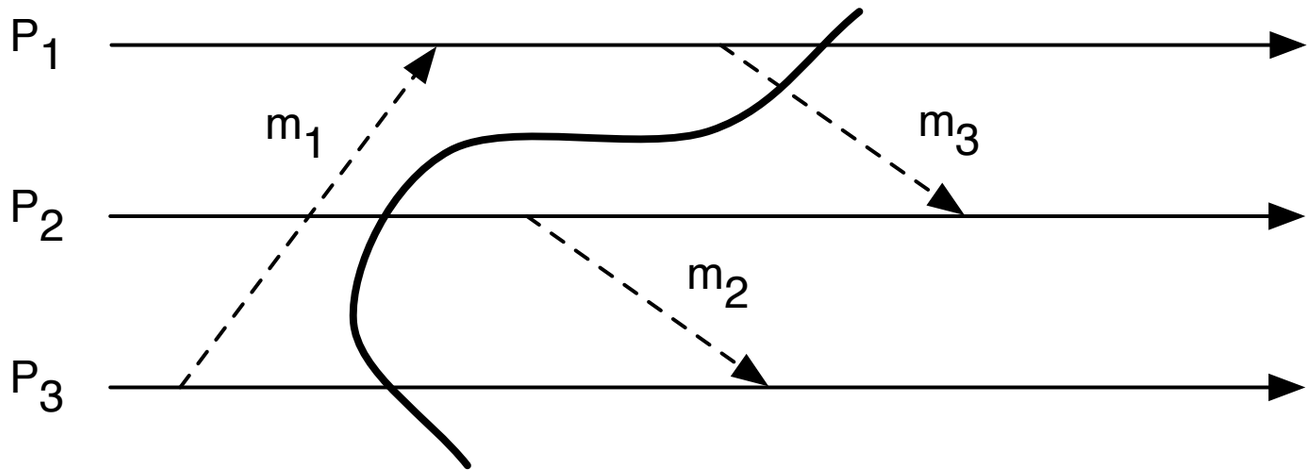
send(m_2) ∉ Snapshot(P_2)
receive(m_3) ∉ Snapshot(P_2)

send(m_1) ∈ Snapshot(P_3)
receive(m_2) ∉ Snapshot(P_3)

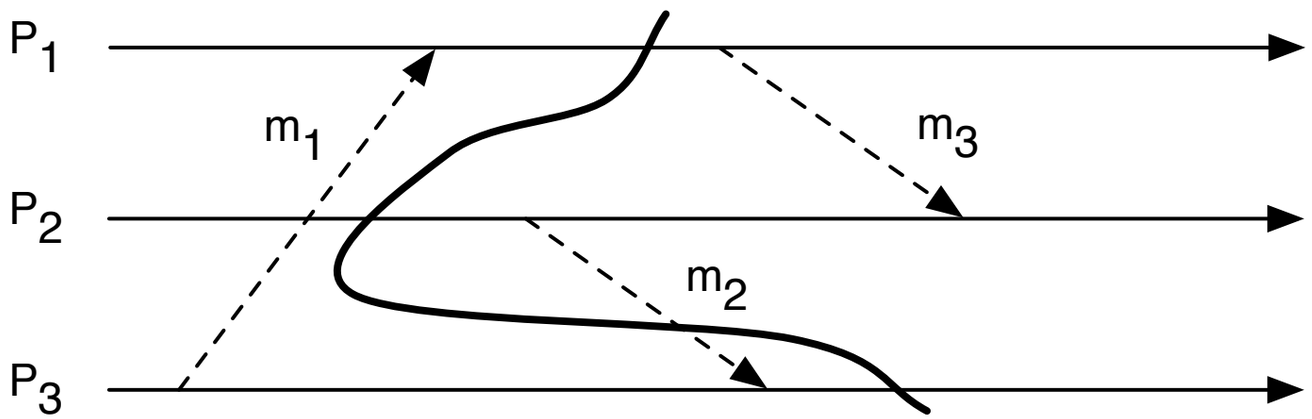


Konsistente Schnitte

- Ein konsistenter Schnitt ist einer, bei dem die Konsistenz-bedingung für verteilte Schnappschüsse gewahrt ist.
- Beispiele:



m_3 in der Übertragung → konsistenter Schnitt

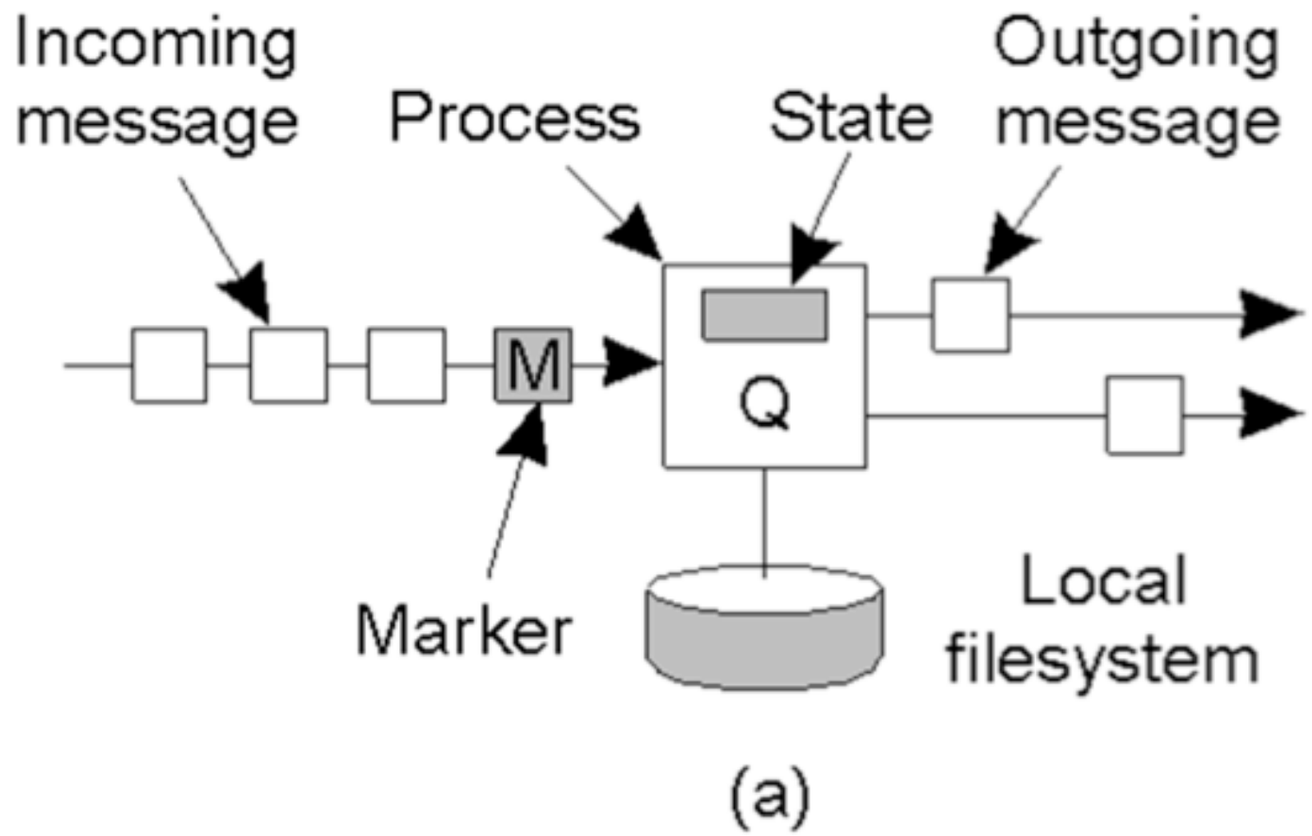


Senden von m_2 fehlt → inkonsistenter Schnitt

Lamport/Chandy-Algorithmus

- Voraussetzung: alle Prozesse sind paarweise über FIFO-Kanäle miteinander verbunden.
- Die Prozesse verständigen sich über Markierungsnachrichten über die Notwendigkeit der Speicherung eines Systemzustands.
- Ein oder mehrere Prozesse starten den Algorithmus, d.h. es können mehrere Schnappschüsse gleichzeitig erstellt werden.
- Das System läuft unterdessen ungehindert weiter.

Prozessmodell



aus: [Tanenbaum, van Steen. *Verteilte Systeme: Grundlagen und Paradigmen*]

Lamport/Chandy-Algorithmus

Für den initiierenden Prozess:

- speichere lokalen Zustand
- starte Aufzeichnungsmodus für alle Eingangskanäle
- sende Markierungen in alle Ausgangskanäle

Lamport/Chandy-Algorithmus

Für jeden Prozess P_i

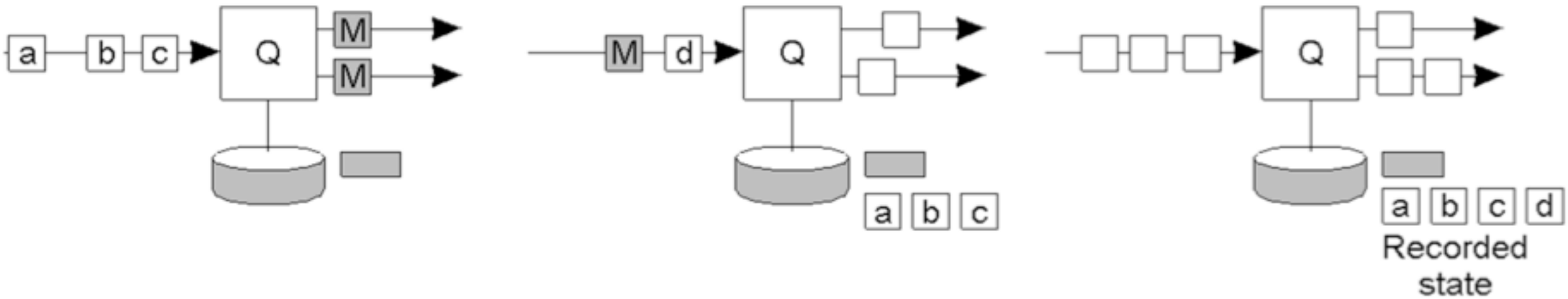
- Wenn eine Markierung über *einen* Eingangskanal c eingeht, dann
  - falls sich P_i nicht im Aufzeichnungsmodus befindet, dann
    - speichere lokalen Zustand von P_i
    - speichere den Zustand von c als leere Liste
    - starte Aufzeichnungsmodus für *alle anderen* Eingangskanäle
    - sende Markierungen in alle Ausgangskanäle

Lamport/Chandy-Algorithmus

- sonst (*P_i befindet sich bereits im Aufzeichnungsmodus*)
  - speichere den Zustand von c als die Liste aller im Aufzeichnungsmodus über c eingegangene Nachrichten
  - beende Aufzeichnungsmodus für c
  - falls keine weiteren Eingangskanäle im Aufzeichnungsmodus,
    - Aufzeichnung des lokalen Schnappschusses vollständig.

- Zusammentragen lokaler Schnappschüsse zu verteiltem Schnappschuss nicht Teil des Lamport-Chandy-Algorithmus.

Ablauf des Algorithmus



(b) (c) (d)

(b) Q initiiert Snapshot: speichert lokalen Zustand, schaltet Aufzeichnen des Eingangskanals an und schickt Marker in alle Ausgangskanäle

(c) Q zeichnet eingehende Nachrichten auf

(d) Nach Empfang eines Markers auf allen Eingangskanälen beendet Q die Kanalaufzeichnung und den lokalen Snapshot

Zusammenfassung (globale Zust.)

- Es ist unmöglich, einen globalen Systemzustand absolut gleichzeitig aufzuzeichnen.
- Der Algorithmus von Lamport und Chandy macht einen Verteilten Schnappschuss.
- Dieser Schnappschuss hat möglicherweise so nie genau als Systemzustand stattgefunden, aber er ist konsistent.

Wahlalgorithmen

Zweck von Auswahlalgorithmen

- In vielen verteilten Algorithmen benötigt man einen Prozess, der eine hervorgehobene Rolle spielt, z.B. als Koordinator, Initiator oder Monitor.
- Zwei Arten von Auswahlalgorithmen:
  1. Bestimmen eines Anführers unter gleichwertigen Prozessen
     → z.B. Bully-Algorithmus, Ring-Algorithmus
  2. Bestimmen des besten Anführers unter verschiedenwertigen Prozessen
     → z.B. Wahlalgorithmus für Ad-Hoc-Netze

- Wichtigstes Ziel: am Ende der Wahl sind sich alle darüber einig, wer der neue Koordinator ist.

Bully-Algorithmus

- **Voraussetzung**
  - Jeder Prozess hat eine ID, die allen anderen Gruppenmitgliedern bekannt ist

- **Initiierung**
  - Prozess P bekommt keine Antwort von Koordinator (Time-Out) → P initiiert Bully-Algorithmus, oder
  - Prozess P kommt neu zur Gruppe hinzu (kann zu Übernahme der Koordinatorrolle führen)

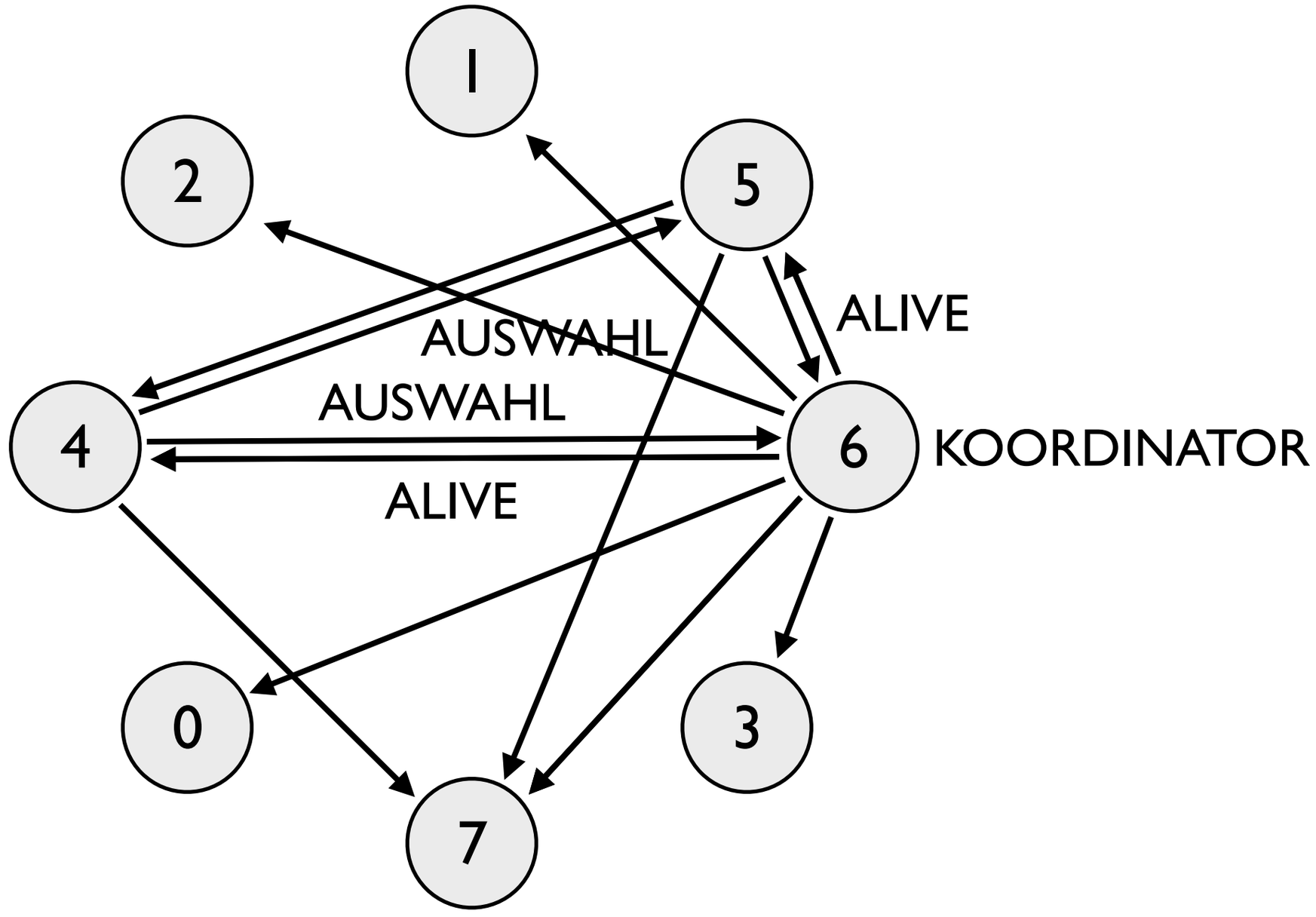
Bully-Algorithmus

- **Algorithmus für Initiator**:
  - P sendet AUSWAHL-Nachricht an alle Prozesse mit höherer Nummer
  - Keine Antwort → P sendet KOORDINATOR-Nachricht an alle Prozesse, startet Koordinatorfunktionalität.
  - mind. 1 ALIVE-Antwort → P wartet bestimmte Zeit auf KOORDINATOR-Nachricht
    - trifft ein → P's Aufgabe erledigt
    - trifft nicht ein → P startet Bully-Algorithmus erneut.

Bully-Algorithmus

- **Algorithmus für jeden anderen Prozess Q**:
  - sobald AUSWAHL-Nachricht von P empfangen
    → antworte P mit ALIVE-Nachricht
    → starte Bully-Algorithmus

Bully-Algorithmus: Beispiel



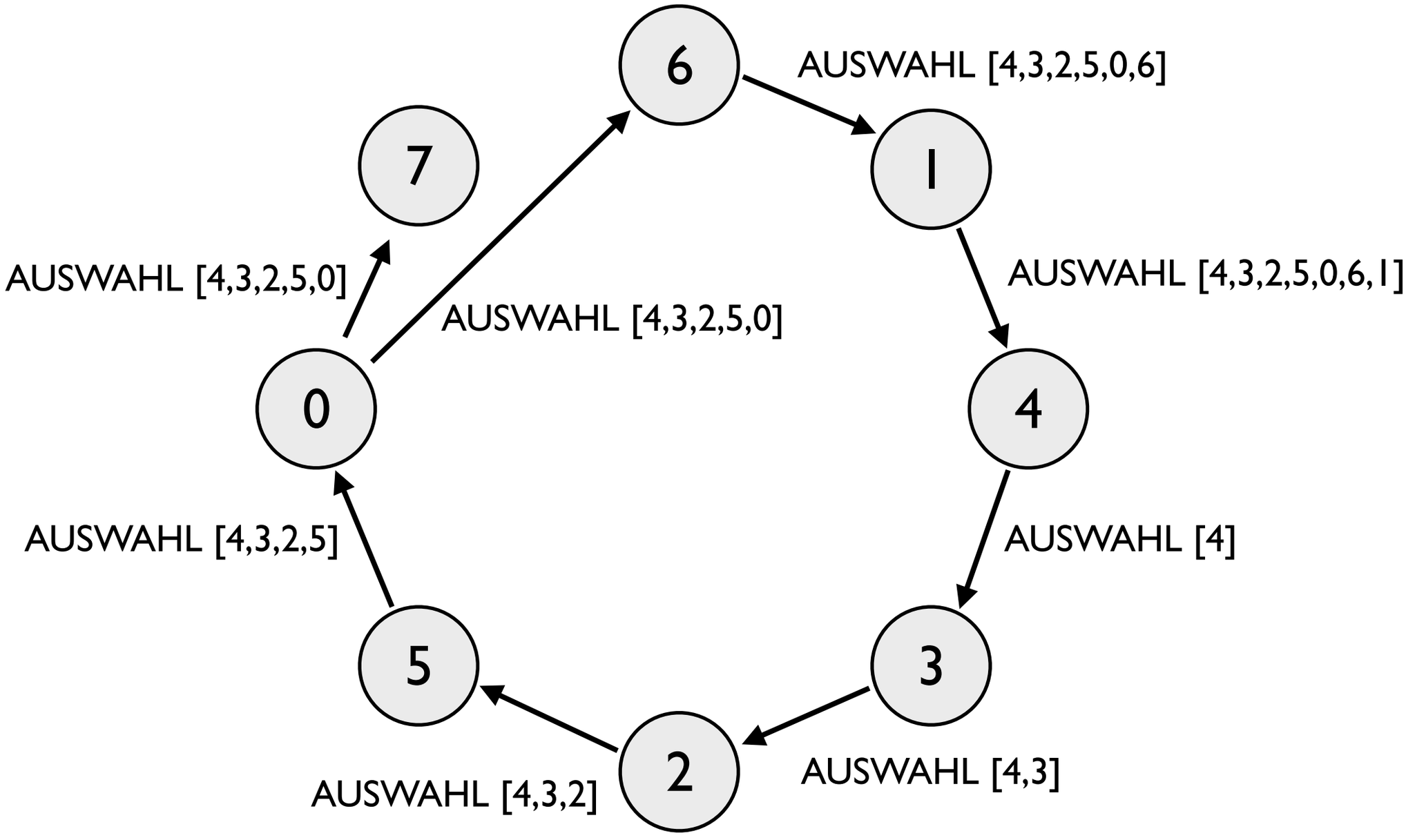
Ring-Algorithmus

- **Voraussetzung**
  - Prozesse sind in zufälliger Reihenfolge in Form eines Rings organisiert, d.h., jeder Prozess besitzt einen Nachfolger, an den er Nachrichten verschicken kann.
- Wenn der Nachfolger eines Prozesses P ausfällt, verwendet P den übernächsten Prozess als neuen Nachbarn.
- **Algorithmus**
  - Wenn ein Prozess feststellt, dass der Koordinator ausgefallen, sendet er AUSWAHL-Nachricht an seinen Nachbarn, in die er sich als ersten einträgt.

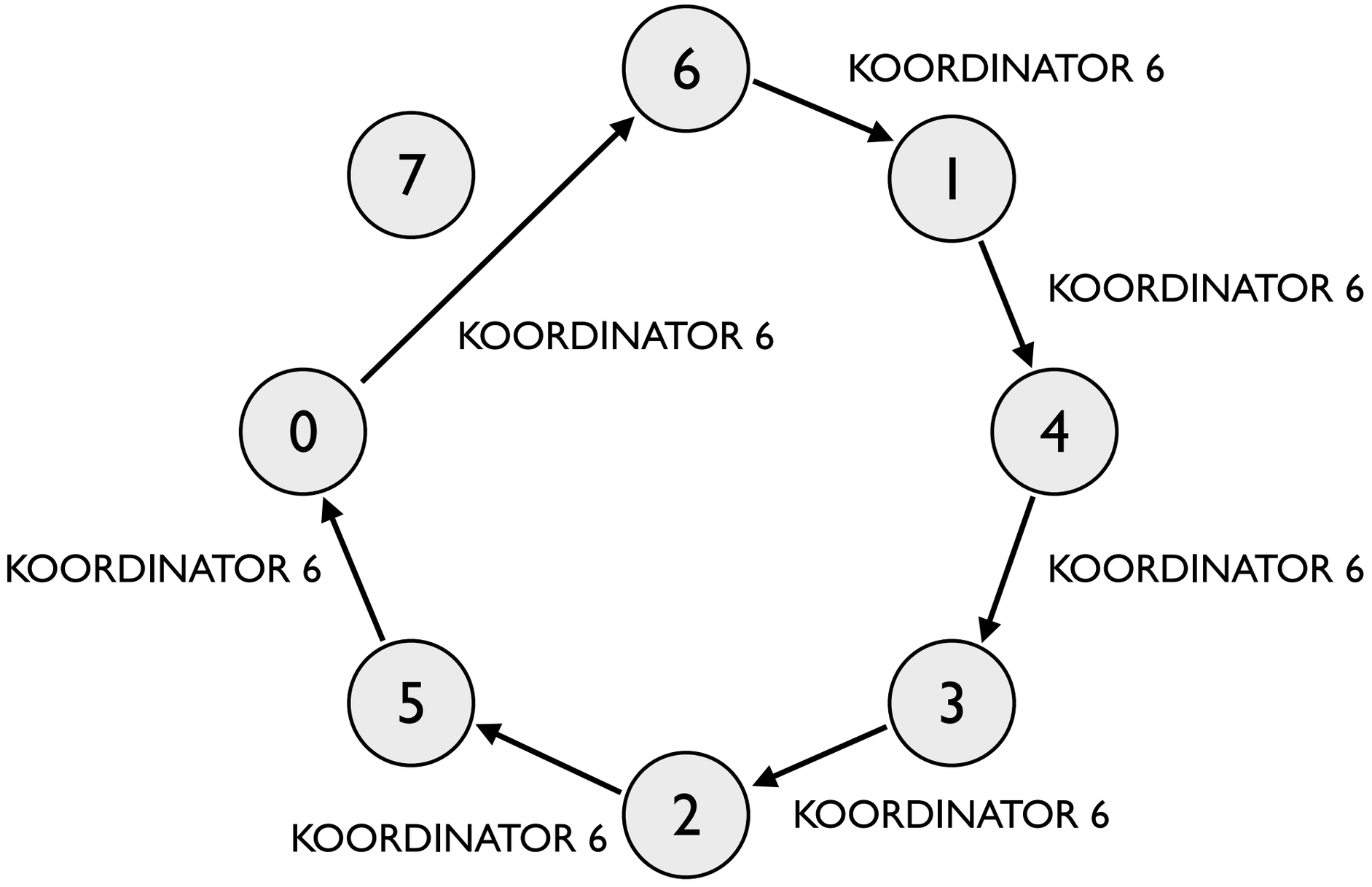
Ring-Algorithmus

- Jeder weitere aktive Prozess fügt sich selbst in Liste ein, gibt modifizierte Nachricht weiter.
- Wenn Nachricht wieder beim Initiator eintrifft, wandelt er sie in KOORDINATOR-Nachricht um, lässt diese wiederum durch Ring kreisen.

Ring-Algorithmus: Beispiel



Ring-Algorithmus: Beispiel



Wahl-Algorithmus für Ad-hoc-Netze

- **Situation**: Netztopologie ändert sich rasch, z.B.
  - drahtlose Netze
  - unstrukturierte P2P-Netze mit hoher Churnrate
- **Ziel**: Finden eines besten Anführers
- **Vereinfachende Annahme**: Knoten bewegen sich während Auswahl vernachlässigbar wenig.
- **Grundidee**: spanne dynamisch Baum auf, propagiere jeweils besten Anführer pro Teilbaum zurück zur Wurzel

Algorithmus

1. Phase: Baum aufspannen

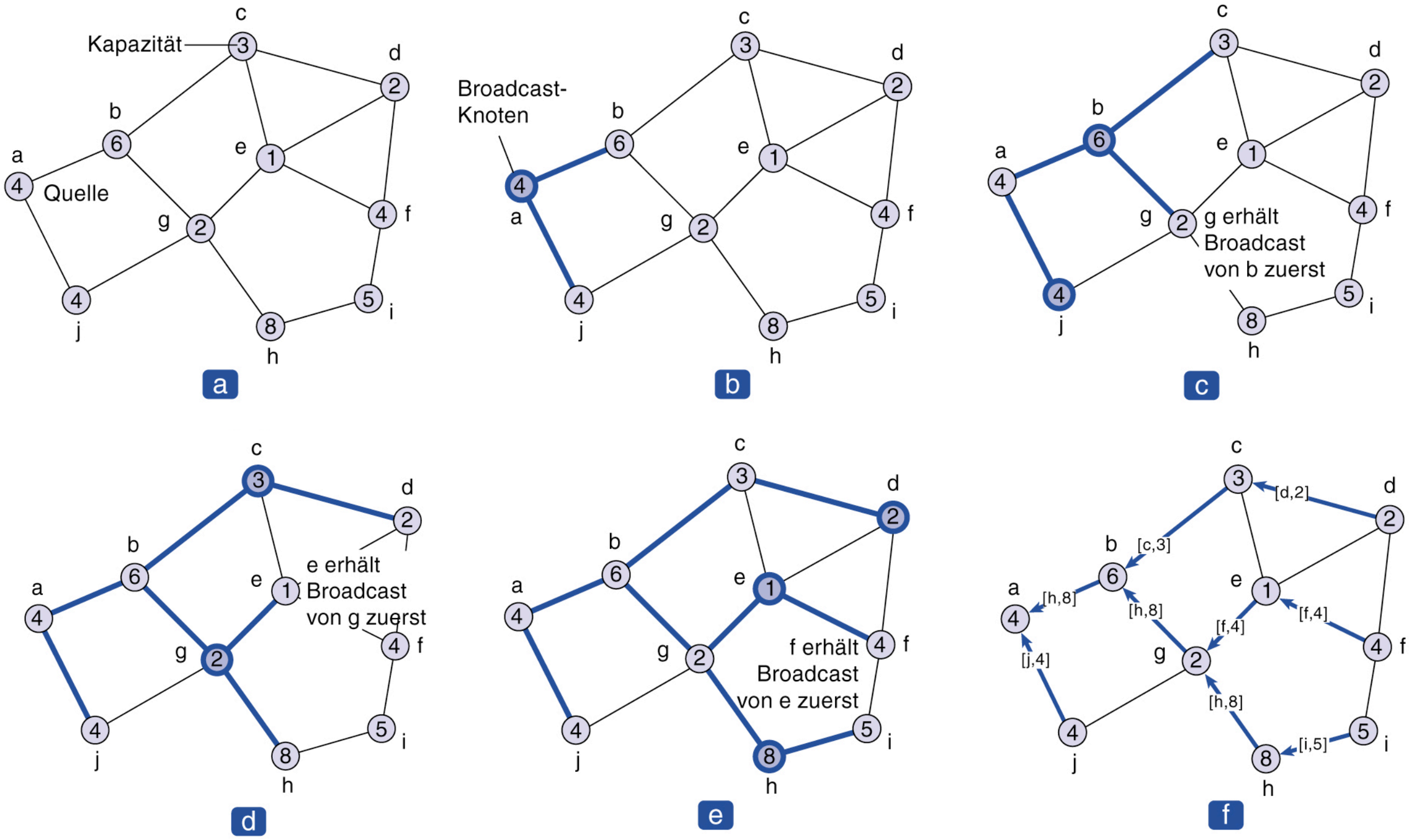
- Ein Knoten, QUELLE, sendet AUSWAHL-Nachricht an alle direkten Nachbarn (z.B. Knoten im Funkbereich)
- Knoten K empfängt erste WAHL-Nachricht:
  - Sender = Elterknoten
  - K sendet AUSWAHL-Nachricht an alle seine Nachbarn außer Elterknoten
- K empfängt weitere WAHL-Nachricht → Empfang quittieren.

Algorithmus

2. Phase: Ergebnis zur Wurzel propagieren

- Knoten K erhält nur Quittungen:
  - K ist Blattknoten
  - K gibt seine wahlrelevanten Eigenschaften (z.B. Ressourcen) zurück an Elterknoten
- Ansonsten:
  - K ist innerer Knoten
  - K wählt aus allen Kindern und sich selbst den besten Anführer und gibt ihn an Elternknoten zurück.

Beispiel



aus: [Tanenbaum, van Steen. *Verteilte Systeme: Grundlagen und Paradigmen*]

Zusammenfassung (Wahlalgorithmen)

- **Bully-Algorithmus**
  - jeder Prozess kommuniziert mit jedem
  - *stärkerer* Prozess bekommt AUSWAHL-Nachricht von *schwächerem* → stärkerer übernimmt Auswahlprozess
- **Ring-Algorithmus**
  - begrenzte Kommunikation entlang Ring
  - erfordert, dass übernächster Nachfolger gefunden werden kann
- **Wahl in Ad-hoc-Netzen**
  - spannt dynamisch Erreichbarkeitsbaum auf
  - hierarchische Auswahl des besten Anführers