Verteilte Systeme

Verteilte Algorithmen

Was sind Verteilte Systeme?

- A distributed system is one in which components located at networked computers communicate and coordinate their actions only by passing messages. [Coulouris]

- Verteiltes System (distributed system) :
  Prozessoren bzw. Prozesse haben keinen gemeinsamen Speicher und müssen daher über Nachrichten kommunizieren. [Löhr]

Wie charakterisiert man VS?

- Spezifikation der Prozesse (vgl. Spezifikation eines ADT)
  - Eingabe/Ausgabe verhalten
- Beziehungen zwischen Prozessen
  - Architektur
  - Kommunikationsparadigmen

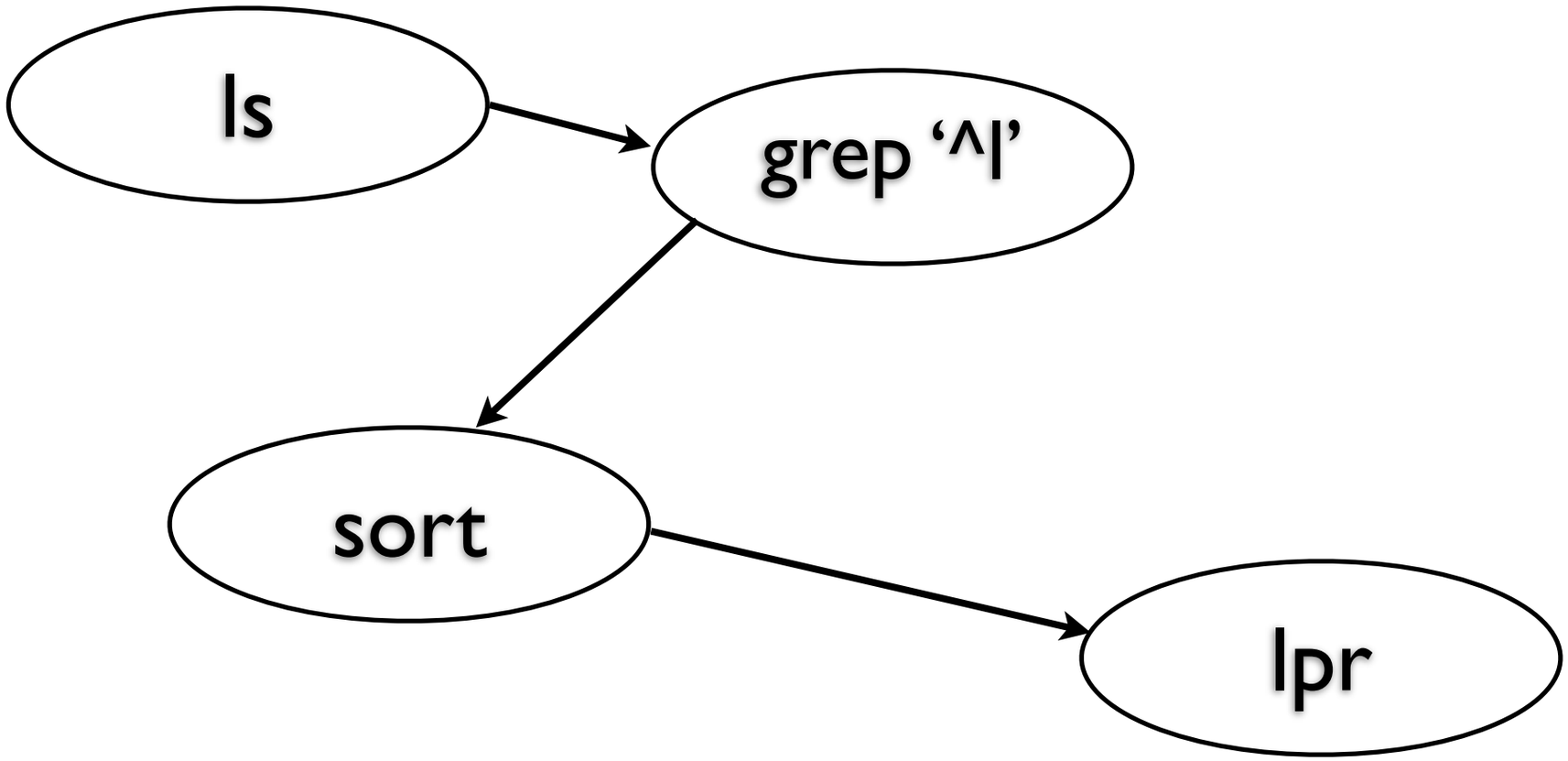
Kommunikationsparadigmen

- Datenfluß-Architektur
- Client/Server-Architektur
- Verteilter Algorithmus

Datenfluß-Architektur

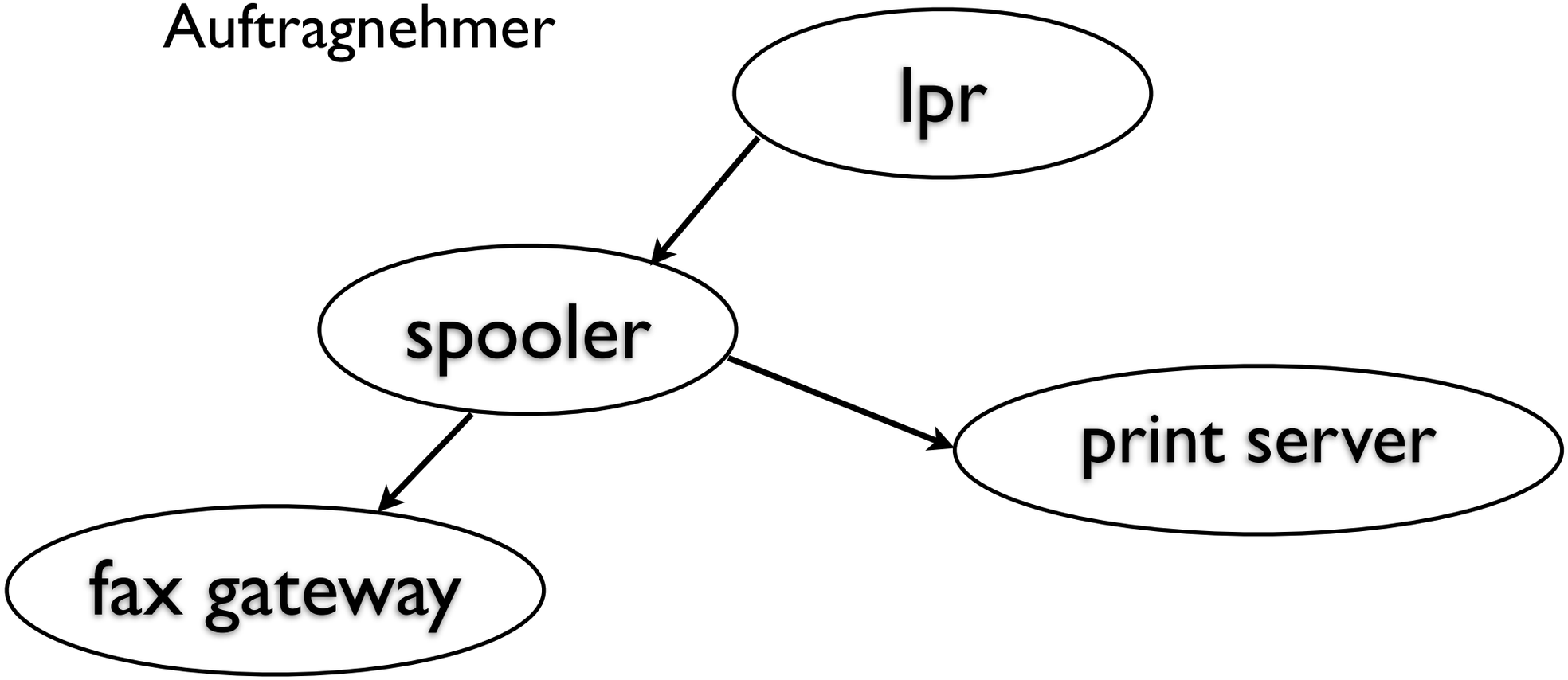
dataflow architecture, “pipes and filters”

- keine ausgezeichneten Rollen
- keine Erwartungen an Kommunikationspartner



Client/Server-Architektur

- Rollen: Auftraggeber und Auftragnehmer
- Einseitige Erwartung des Auftraggeber an den Auftragnehmer



Verteilter Algorithmus

Prozesse kooperieren zwecks Erreichung eines gemeinsamen Ziels

- in bestimmten, vereinbarten Kommunikationsmustern

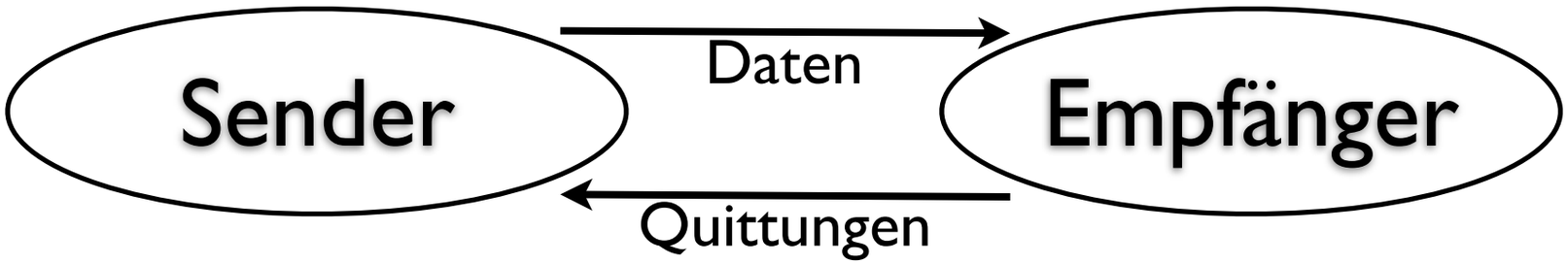
  ➜ Protokoll

- in verschiedenen oder gleichen Rollen

  ➜ z.B. Cluster head

Beispiele:

- Alternating Bit Protokoll zur zuverlässigen unidirektionalen Nachrichtenübertragung über einen unzuverlässigen bidirektionalen Kanal



- Auswahl eines Anführers/Koordinators

Verteilte Systeme

Kausalität und Ordnung von Ereignissen

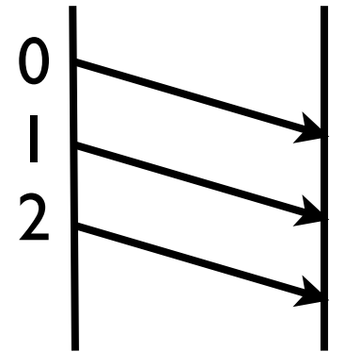
Kausalität und Ordnung von Ereignissen

- Verteilte Systeme haben
  - keinen gemeinsamen Speicher
  - keine gemeinsame Zeit
- Absolute Zeitpunkte sind verteilt nur schwer abschätzbar
- relative Zeitpunkte einzelner Ereignisse können wichtig sein (push vor pop – oder– pop vor push ?)

Sende- und Empfangsreihenfolge

Bei einem Sender/Empfänger Paar:

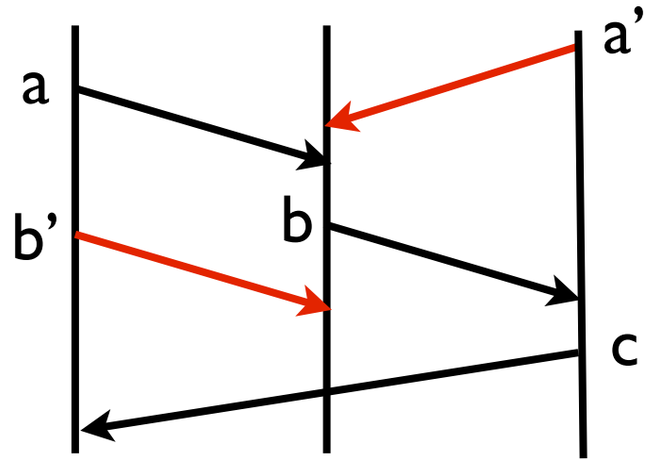
leicht, durchnummerieren der Nachrichten genügt



Bei mehreren Sender/Empfänger:

wie durchnummerieren?

Wie können Kausalketten
“Wurde b nach dem Eintreffen von a versendet oder nicht”
abgebildet werden



Kausale Ordnung von Ereignissen

Def: Ereignis (event)

- lokal – Prozessinterne Aktivität (z.B. Zuweisung)
- send – Senden einer Nachricht
- recv – Empfangen einer Nachricht

Def: Ereignismenge E, Menge nicht näher spezifizierter Ereignisse

Annahmen:
- Prozesse sind sequentiell
- Es gehen keine Nachrichten verloren

Kausale Abhängigkeit

Def: Für zwei Ereignisse a,b ∈ E gilt: b ist von a <u>Kausal abhängig</u>: a ➔ b, wenn eine der Bedingungen zutrifft:

1. a und b gehören zu einem Prozess und werden in der Reihenfolge ausgeführt

2. a ist das Sender einer Nachricht n,
   b ist das Empfangen der Nachricht n

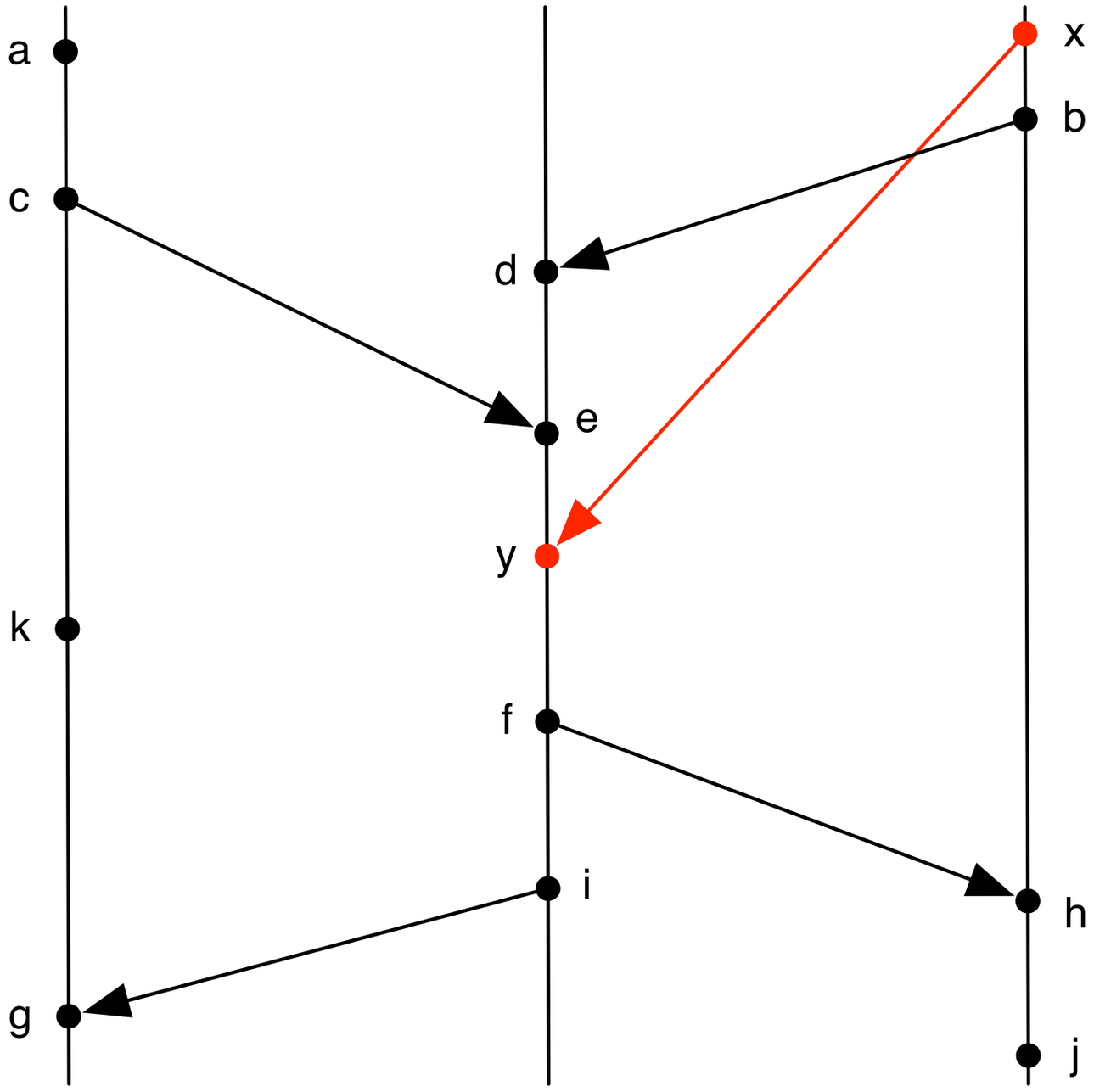
3. es gibt ein Ereignis x für das gilt:
   a ➔ x und x ➔ b (Transitivität)

Kausale Abhängigkeit

Korollar: Wenn wir ➔ als “gleich oder vor” definieren,
ist die Kausalordung ➔ Partielle Ordnung auf E

Def: Zwei Ereignisse a,b sind unabhängig a ⇹ b
(Nebenläuftig, Concurrent), wenn
weder a ➔ b noch b ➔ a gilt.

Kausale Abhängigkeit



es gilt:

a → c → k → g

a → c → e

c → h

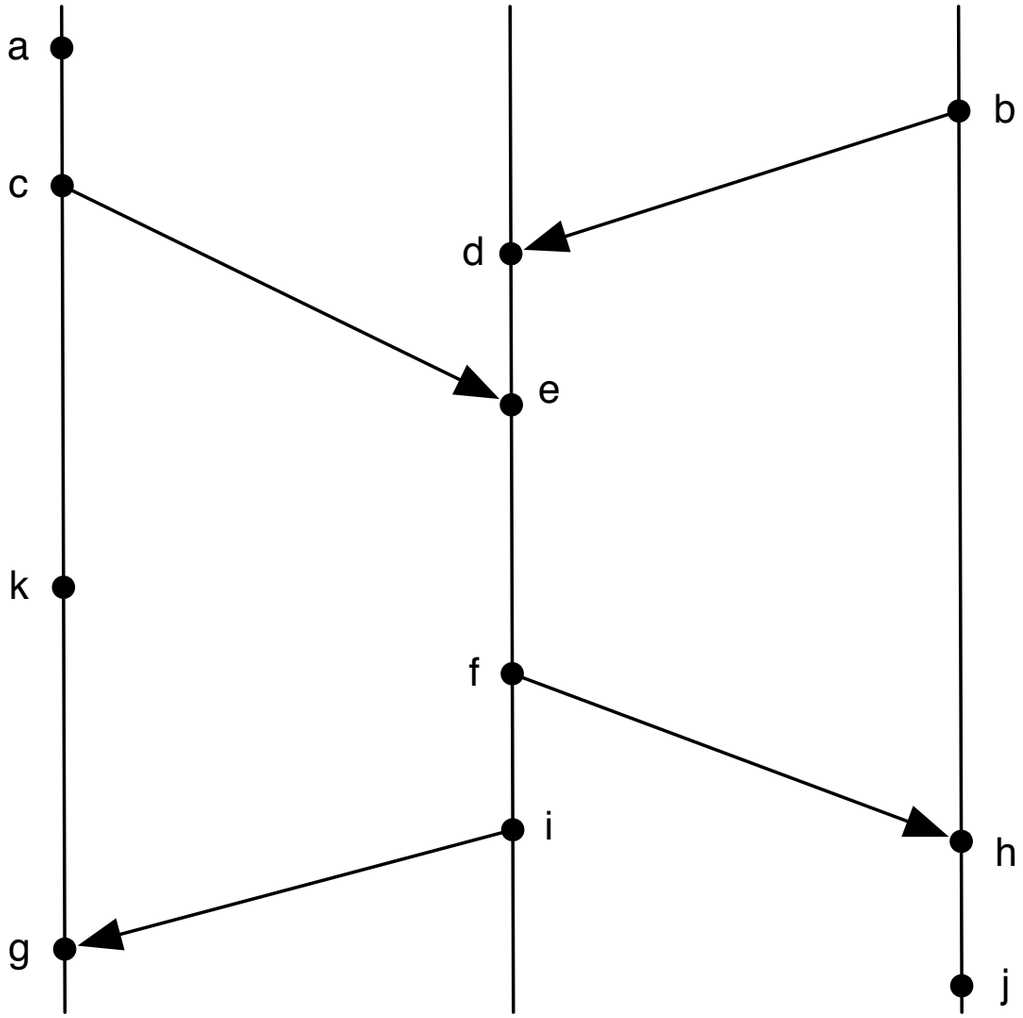
es gilt nicht:

a → d

k → j

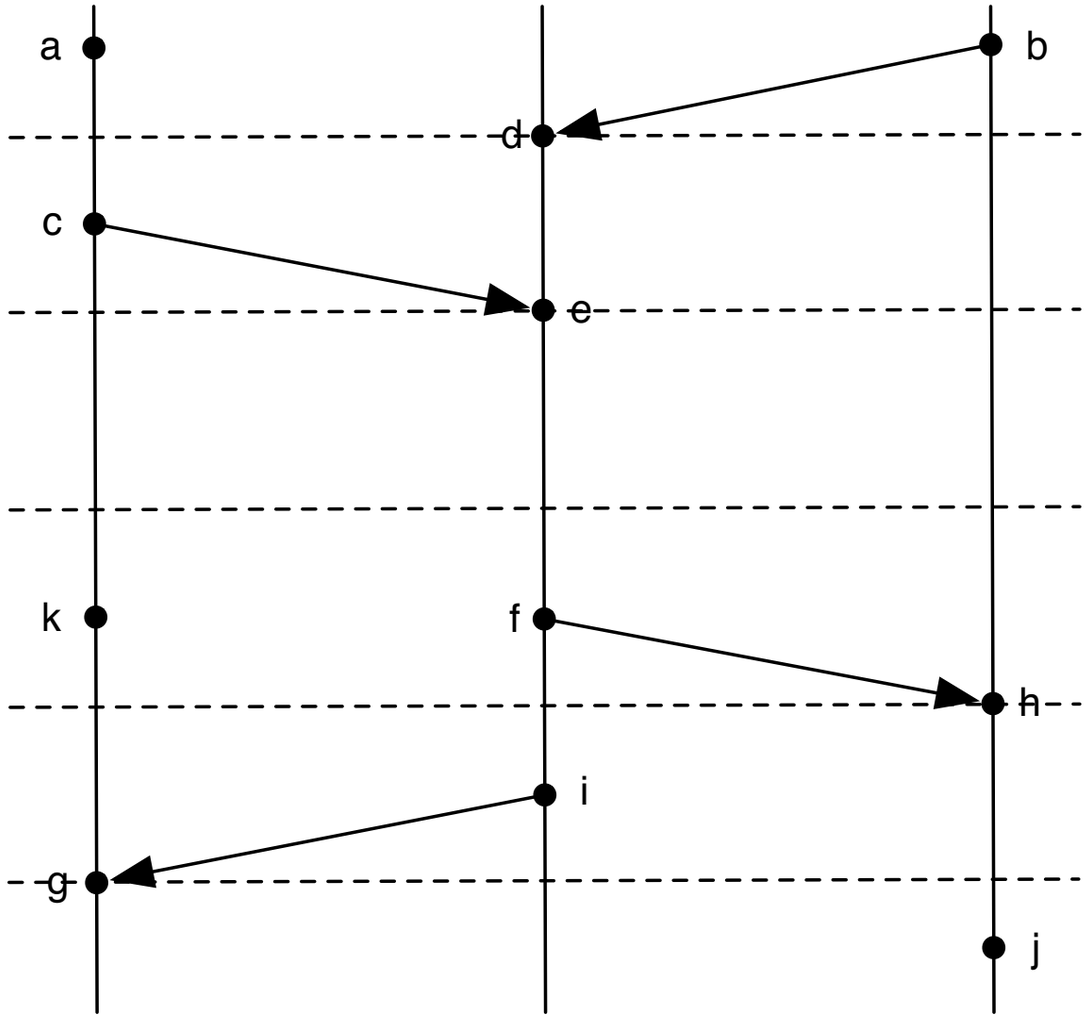
aus x → b, d → y folgt dass die Nachrichten nicht FIFO übertragen wurden!

Asynchrone vs. Synchrone Verteilte Systeme



Asynchroneres VS

Ereignisse können zu beliebigen Zeitpunkten Eintreten, kein konsistenter Zustand



Synchroneres VS

Ereignisse sind in Runden aufgeteilt. Konsistenter Zustand am Ende einer Runde

Verteilte Systeme

Logische Uhren

Logische Uhren

- Kausalität - “happend before” - wird im täglichen Leben meist mit Uhrzeiten verbunden.
- Idee: Eine numerische Repräsentation für kausale Abhängigkeit verwenden.

Jedem Ereignis e ∈ E wird eine Zeit C(e) ∈ T zugeordnet

Die Zeiten sind partiell geordnet und die Ordnung sollte isomorph zur kausalen Ordnung sein: (E,➔) ≡ (T,<)

Die Abbildung C: E → T heißt logische Uhr

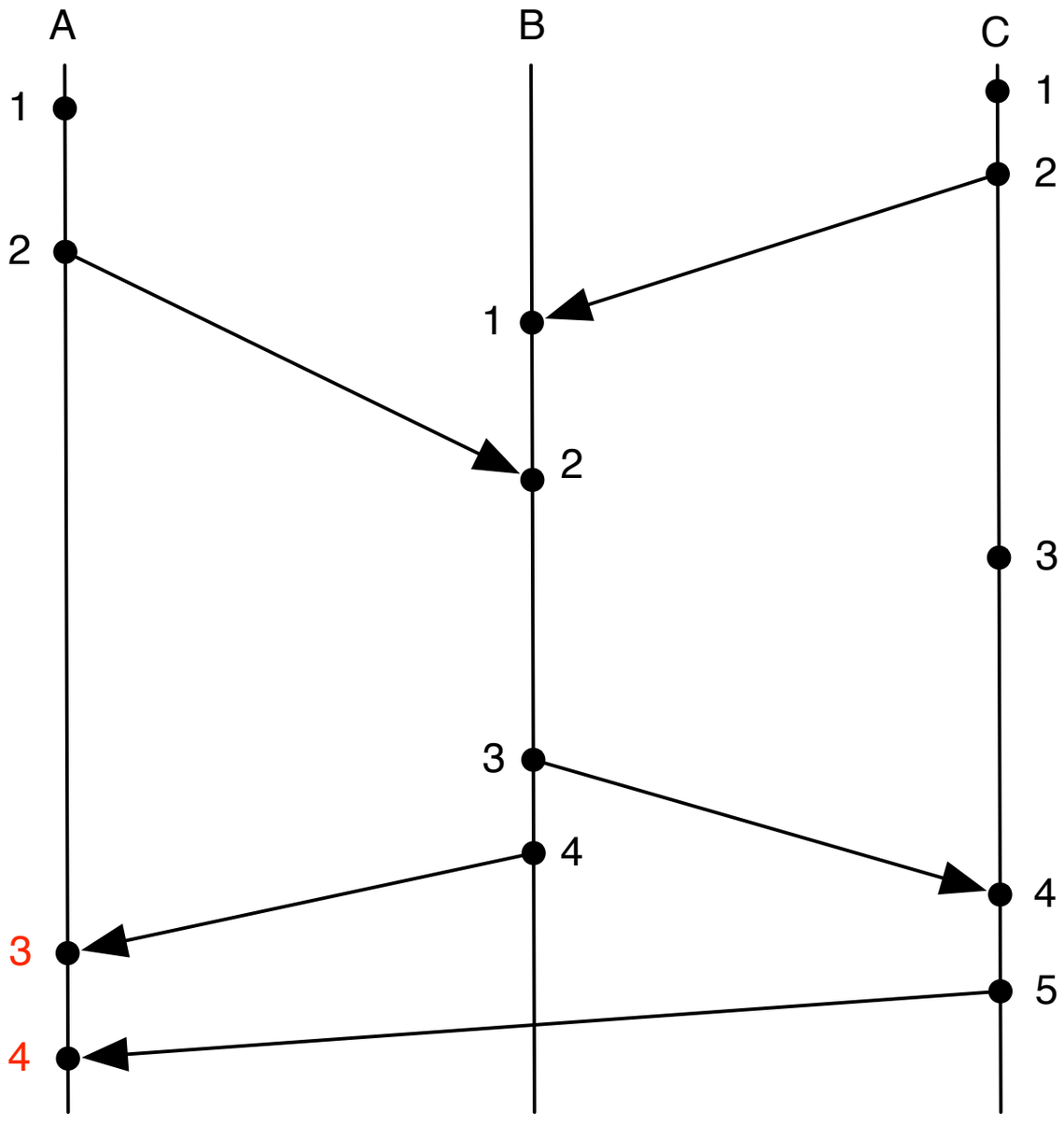
Skalare Zeit [Lamport 1978]

T = natürliche Zahlen

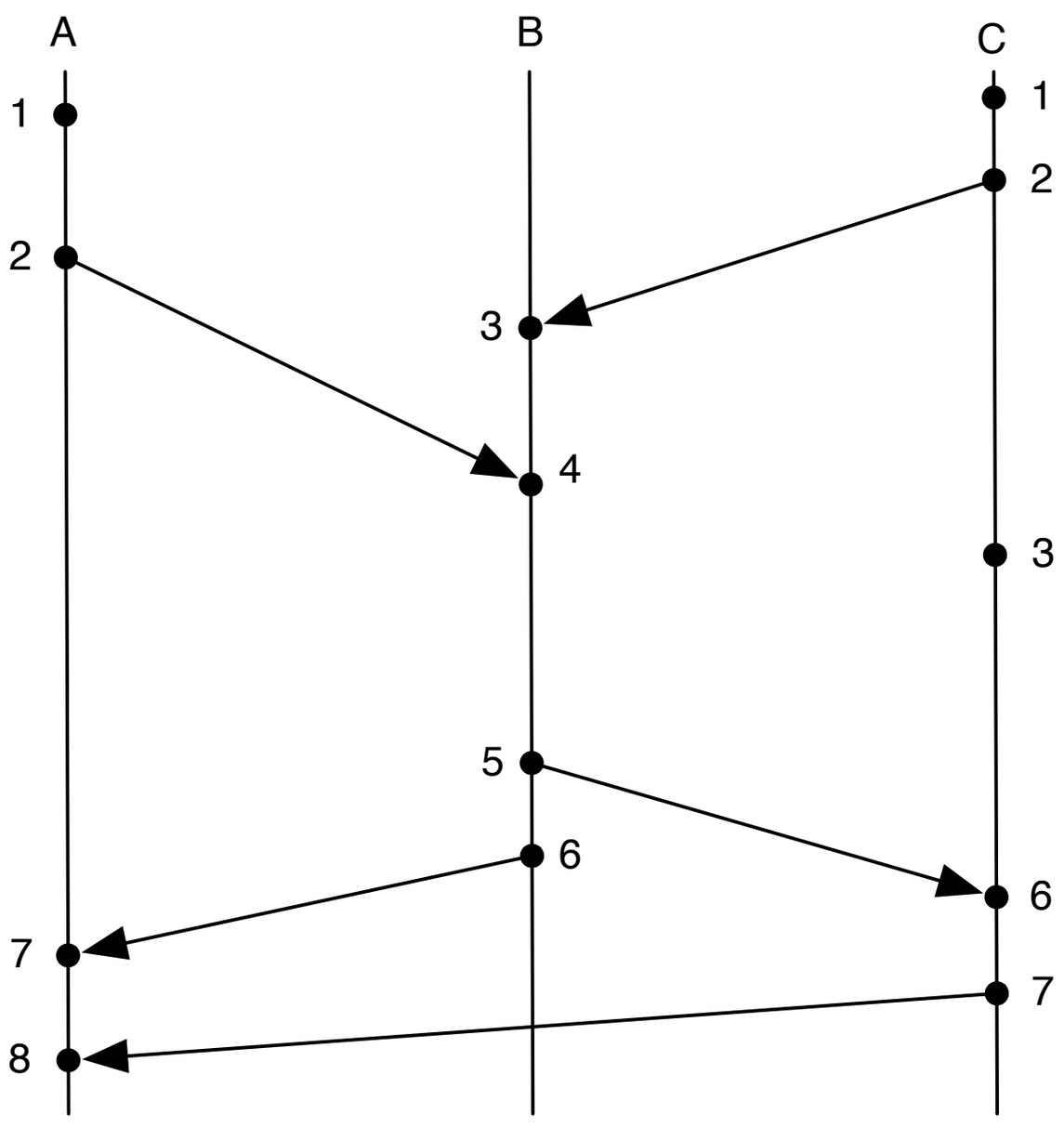
Initial: c = 0 für jeden Prozess

- Jede Nachricht wird mit einem Timestamp t = c versehen
- Vor jedem Ereignis wird c inkrementiert
- Nach jedem (m,t) = recv() wird c = max(c,t+1,c) gesetzt

Skalare Zeit [Lamport 1978]

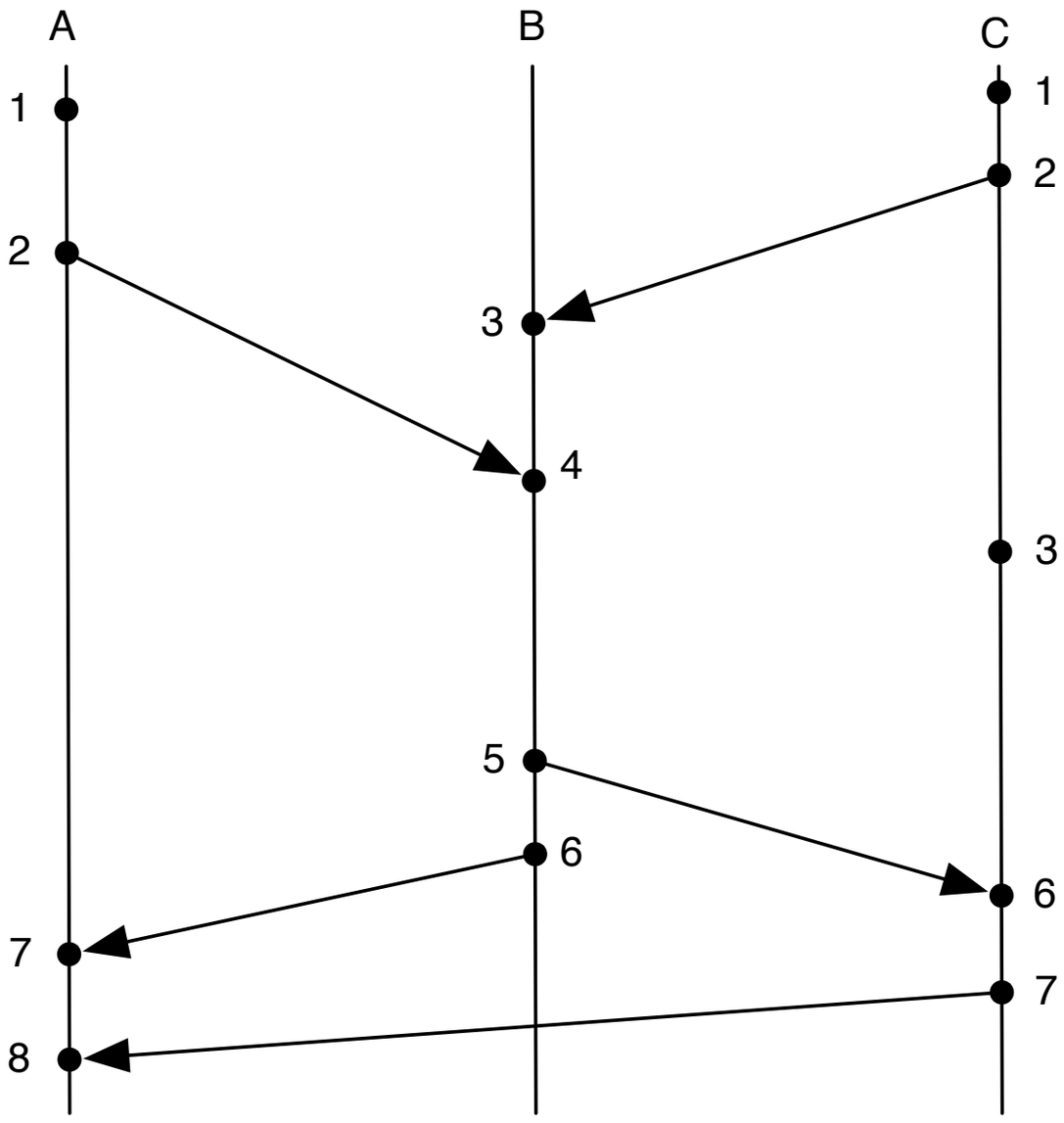


naiver Ansatz

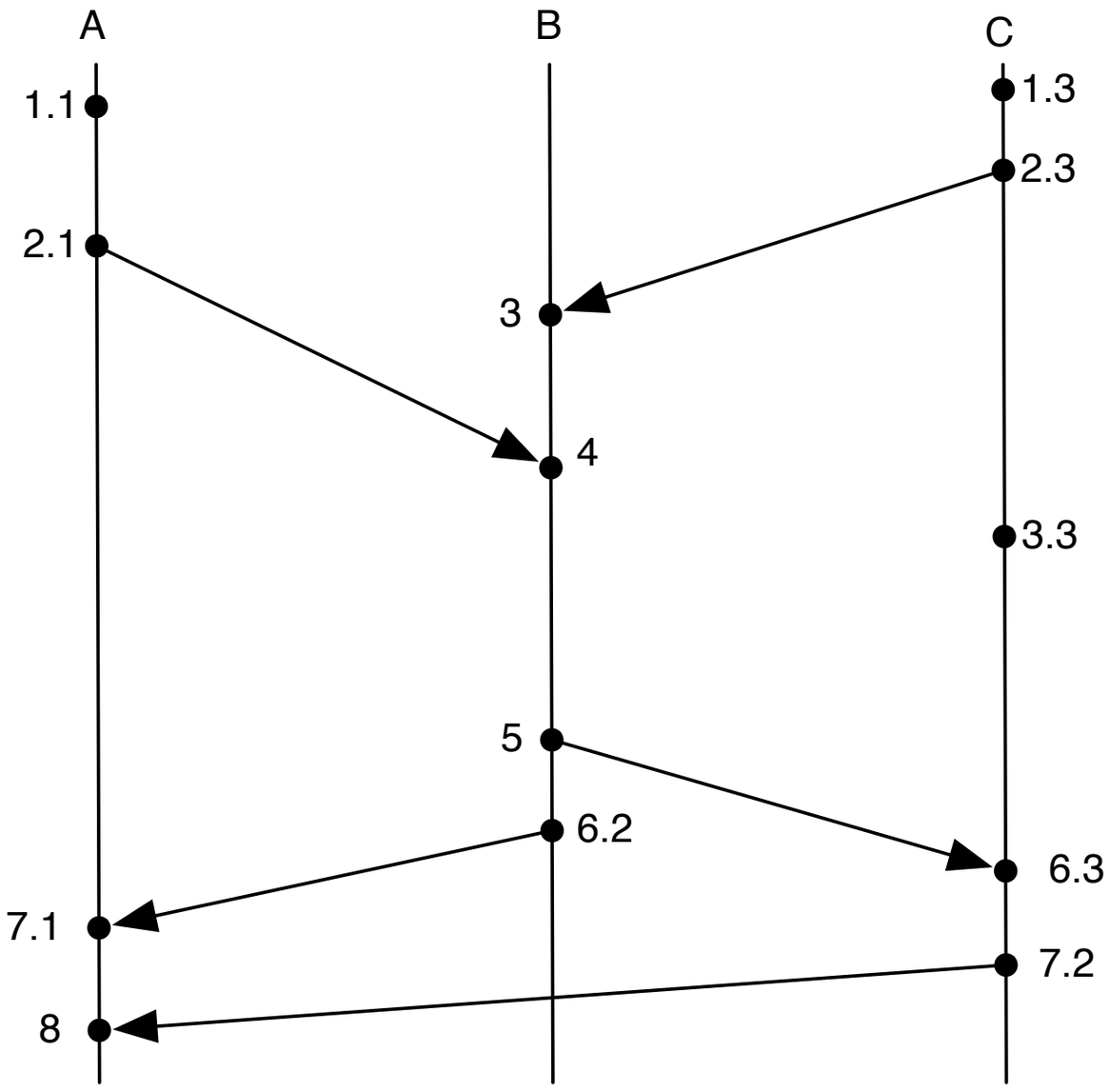


Skalare Zeit

Skalare Zeit [Lamport 1978]



Skalare Zeit bildet nur eine Halbordnung



lässt sich aber zu einer totalen Ordnung erweitern

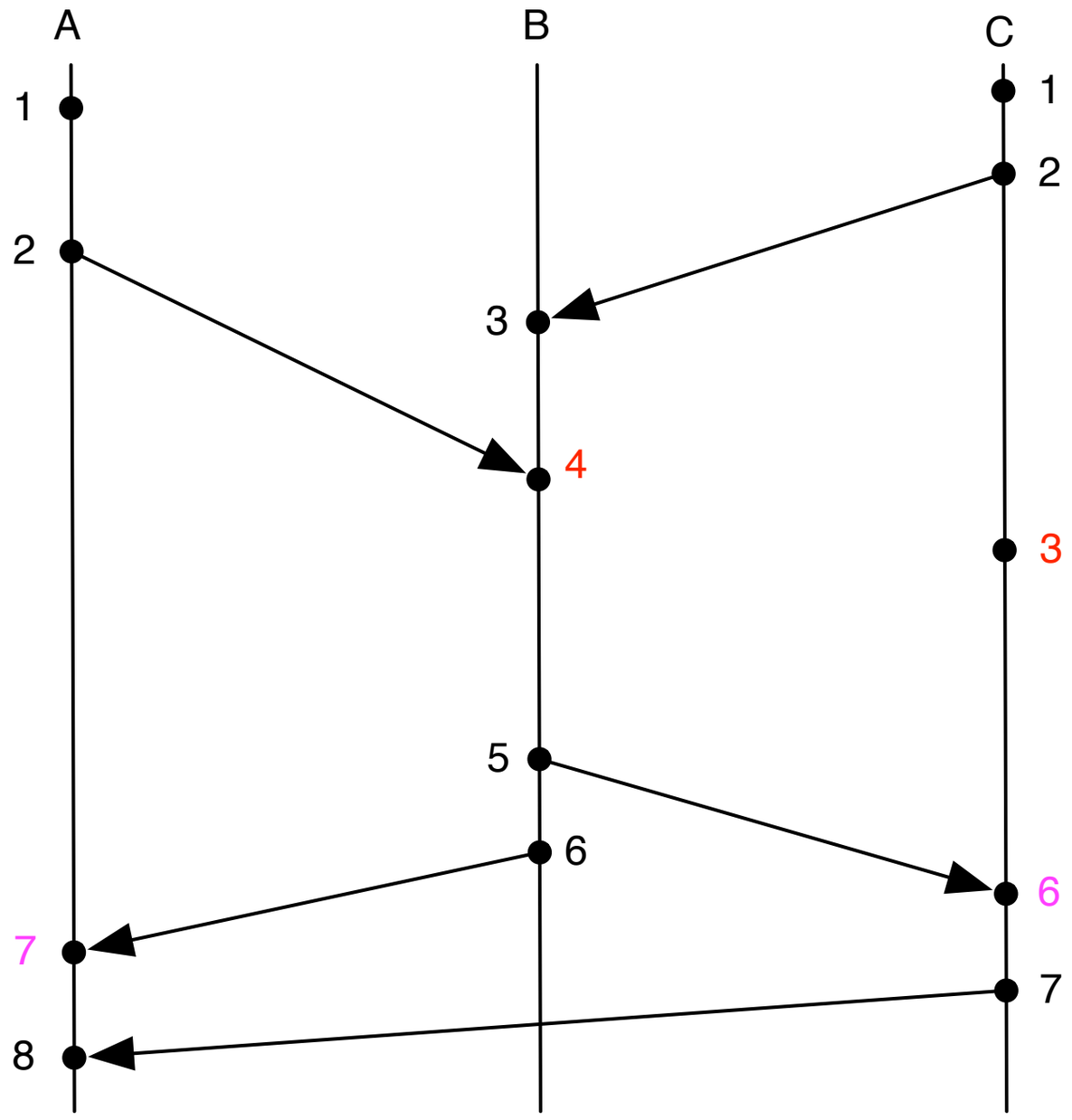
Skalare Zeit [Lamport 1978]

Für beliebige Ereignisse gilt:

$$a \rightarrow b \Rightarrow C(a) < C(b)$$

aber nicht die Umkehrung!

Es ist nicht entscheidbar anahand der Skalaren Zeit, ob Ereignisse unabhängig sind!



Vektorzeit [Fidge, Mattern 1988]

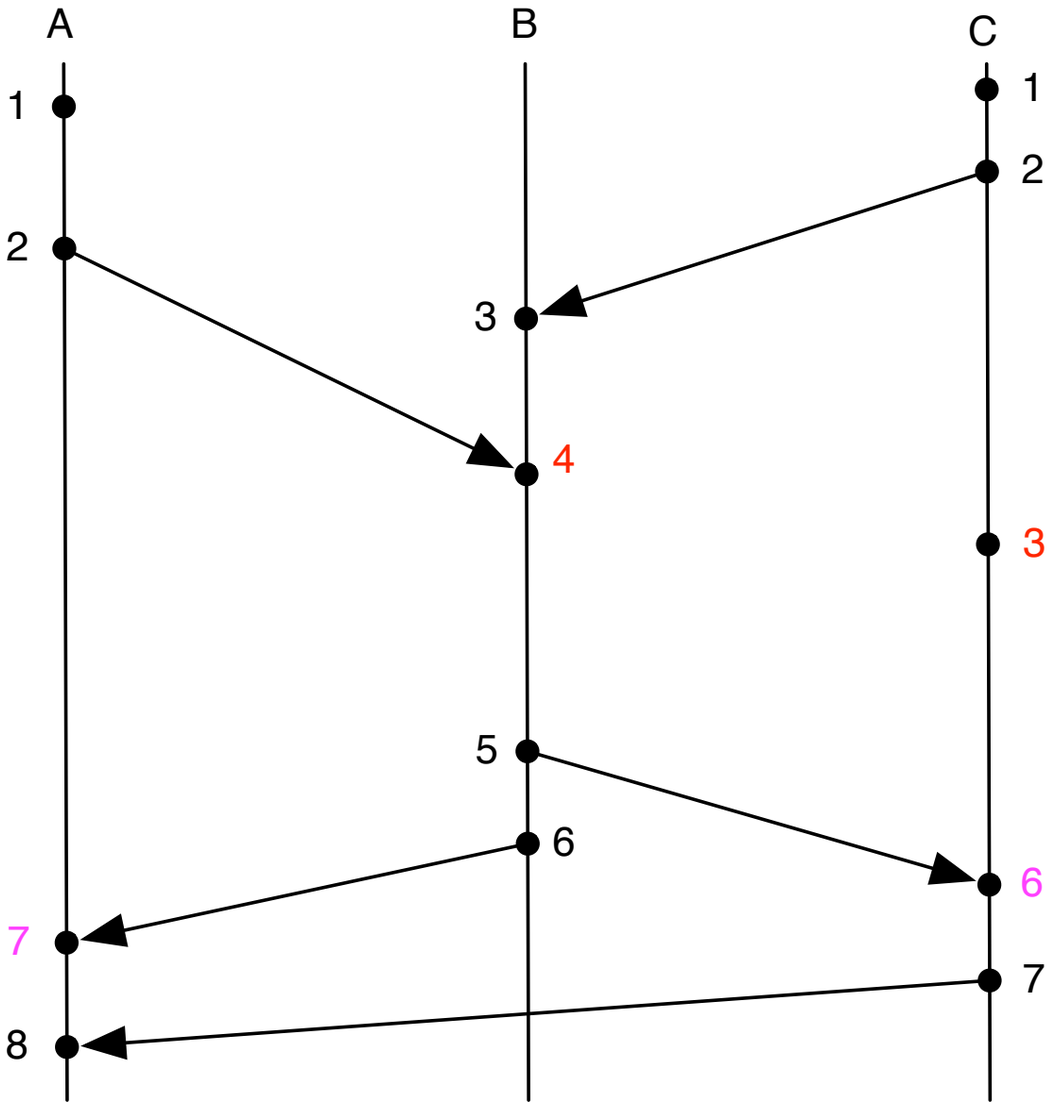
T = n-Tupel natürlicher Zahlen (n = #Prozesse)

$t \leq s$: $t_i \leq s_i \ \forall \ i \in (1..n)$ $\Rightarrow$ Halbordnung

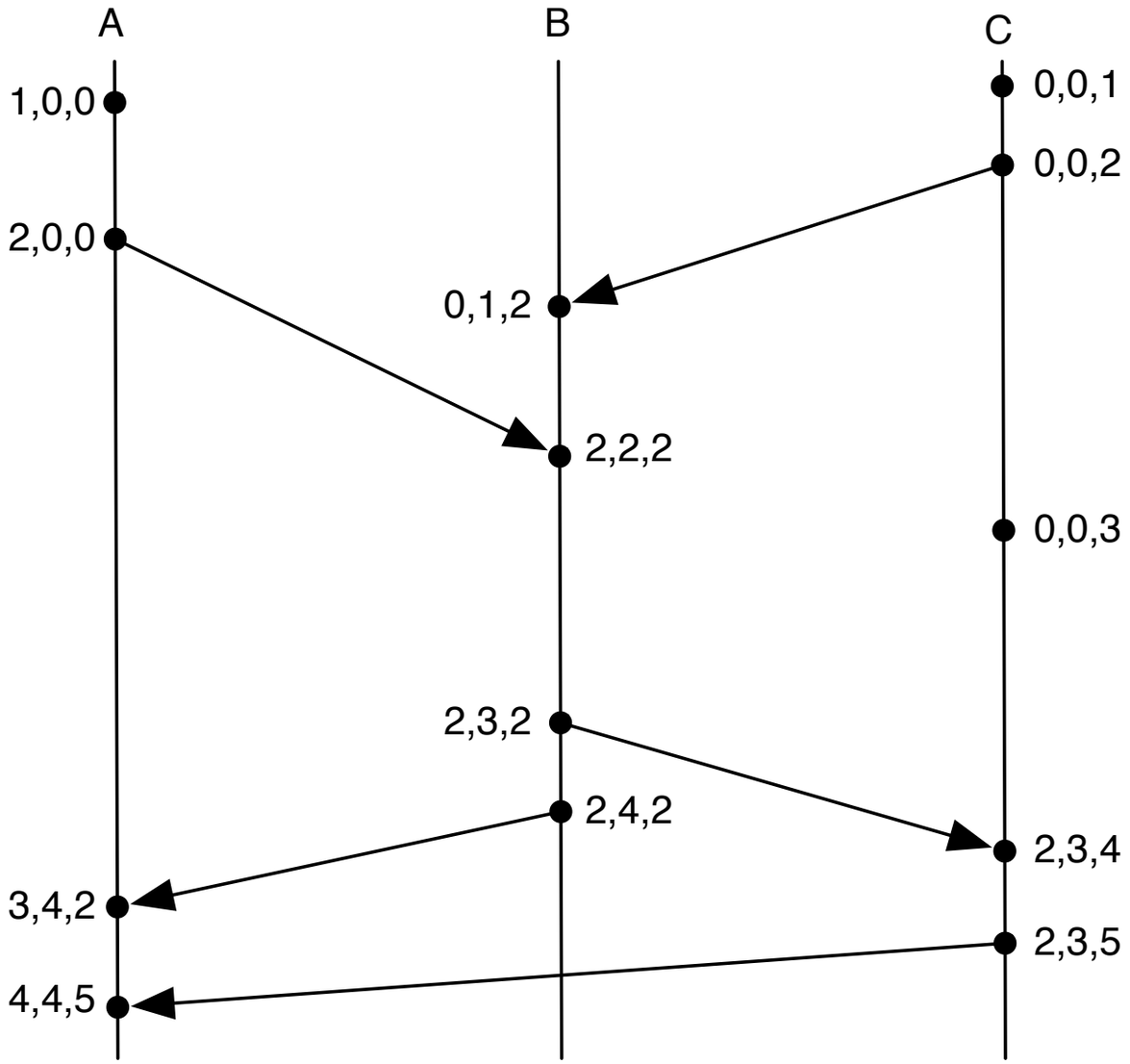
Initial: $c_p = 0$ für jeden Prozess p

- Jede Nachricht wird mit einem Timestamp t = c versehen
- Vor jedem Ereignis wird c_p inkrementiert
- Nach jedem (m,t) = recv() wird $c_i = max(c_i,t_i)$ gesetzt

Skalare Zeit vs. Vektorzeit



Skalare Zeit



Vektorzeit

Skalare Zeit vs. Vektorzeit

Skalare Zeit ist eine logische Uhr, die Ereignisse total ordnet und dabei die kausale Abhängigkeit erhält:

$$a \rightarrow b \Rightarrow C(a) < C(b)$$

Vektorzeit erhält zusätzlich die Information, welche Ereignisse unabhängig sind:

$$a \nrightarrow b \Leftrightarrow C(a) \cong C(b)$$

$$\text{mit } C(a) \cong C(b) := \exists\, i,j \in (1..n): t_i \leq s_i;\ s_j \leq t_j$$

Verteilte Systeme

Auswahlalgorithmen
(leader election algorithms)

Auswahlalgorithmen
(leader election algorithms)

- dienen zur Wahl eines Koordinators einer Gruppe von gleichberechtigten Stationen bei „halbverteilten“ Algorithmen.

- Nach Ausfall des Koordinators erfolgt die Inbetriebnahme eines neuen Koordinators üblicherweise in 3 Phasen:

1. Feststellung des Ausfalls durch andere Station(en)

2. Neuwahl – Einigung auf einen neuen Koordinator

3. Amtsübernahme des neugewählten Koordinators

Auswahlalgorithmen

Behauptung: Falls alle Stationen gleich sind, existiert
kein deterministischer Auswahlalgorithmus,
der das Problem löst.

Vor: Sei A ein System von n identischen Prozessen in
einem Synchronen Ring.

Bew: o.B.d.A haben alle Prozesse exakt einen Startzustand.
- Für eine beliebige Ausführung gilt per Induktion über die
Anzahl der Runden: Am Ende jeder Runde haben alle Prozesse
den selben Zustand.

⇒ Falls ein Prozess im Zustand “Koordinator” ist, sind es
alle anderen auch ⇒ Widerspruch

Bully-Algorithmus [Garcia-Molina 1982]

Vor: - Stationen können ausfallen
- Stationen sind linear geordnet
- Jede Station kann jede andere per ID adressieren
- Zuverlässiger FIFO Unicast

Ziel: Die verbliebene Station mit der höchsten ID soll Koordinator werden.

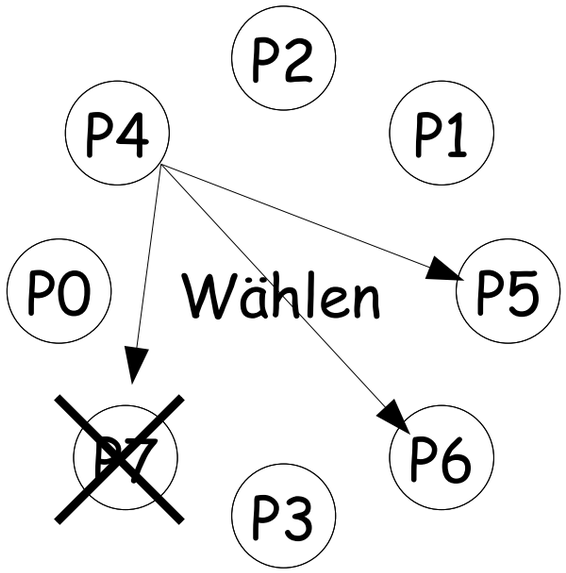
⇒ Koordinator-Wechsel nur bei Ausfall des aktuellen Koordinators oder wiederkehren einer Station mit höherer ID.

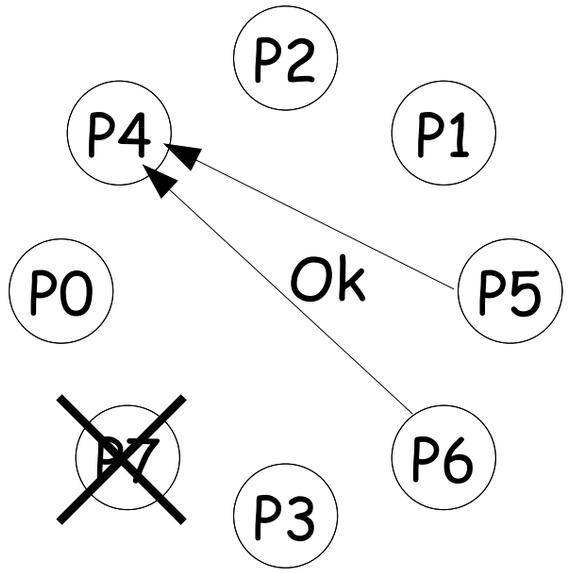
Bully-Algorithmus [Garcia-Molina 1982]

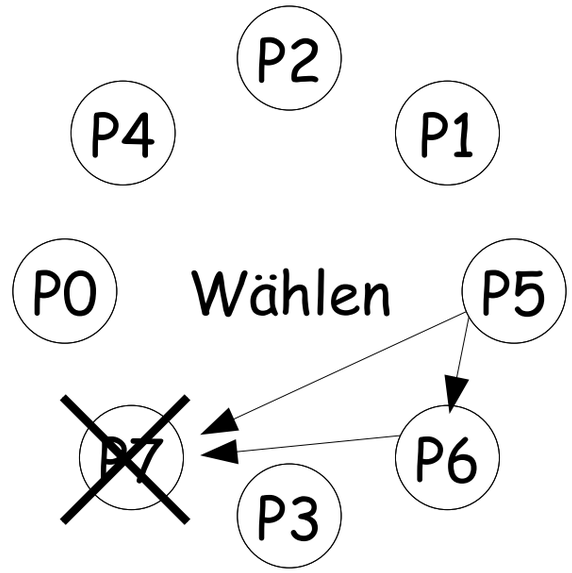
Ablauf:

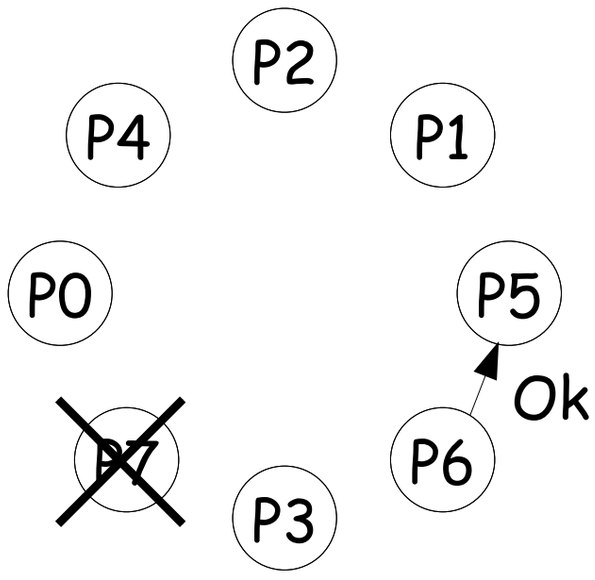
1. Beliebige Station erkennt den Ausfall des Koordinators
2. Die Station fordert alle ihr bekannten Stationen mit höherer ID zur Wahl auf.
3. Alle funktionsfähigen Stationen quittieren die Nachricht mit OK und fordern ihrerseits zur Wahl auf.
4. Warten auf die Quittungen:
   - Wenn kein OK eintrifft (=Station mit größter ID), erklärt sich die Station selbst zum Koordinator und teilt das Ergebnis den Stationen
   - sobald ein OK eintrifft wartet die Station auf das Wahlergebnis des Koordinators

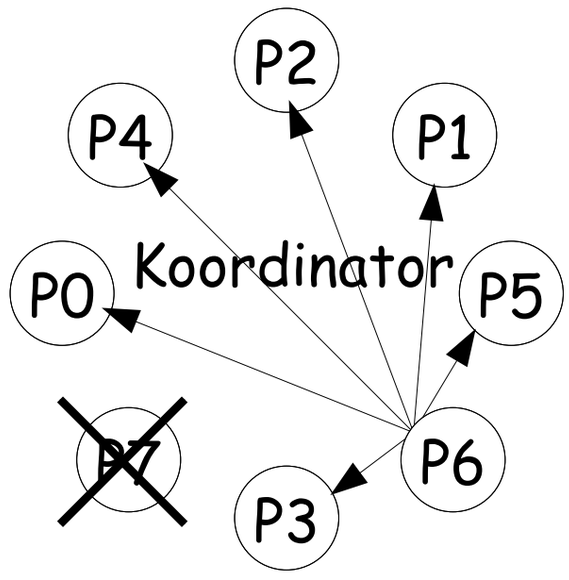
Bully-Algorithmus [Garcia-Molina 1982]











Bully-Algorithmus [Garcia-Molina 1982]

Kommunikationsaufwand:

Bester Fall:

- Station n-1 bemerkt den Ausfall
  - n-2 Ergebnismeldungen
  - O(n)

Ungünstigster Fall:

- Station 1 bemerkt den Ausfall von Station n
  - n(n-1)/2 Wahlaufrufe
  - (n-1)(n-2)/2 Quittungen OK
  - n-2 Ergebnismeldungen

$$\Rightarrow O(n) \leq K(n) \leq O(n^2)$$

Übungsaufgabe zum 5/6.5.2011

Lesen Sie die Papers von Lamport und Fidge.

- Wie verhalten sich Skalare Zeit und Vektorzeit, wenn das Medium nicht FIFO garantiert?

- In welchen Fällen ist es sinnvoll unabhängige Ereignisse zu erkennen.

- Diskutieren Sie die Bedeutung von diskreter vs. kontinuierlicher Zeit.