

# Verteilte Systeme

## 8. Verteilte Terminierung

### Verteilte Terminierung

- Im sequentiellen Fall stellt sich das Problem nicht
- Zwei Terminierungsvarianten
- Kommunikationsorientierte Terminierung
  - Beispiel
    - Alle Prozesse sind passiv
    - Keine Nachrichten sind unterwegs
  - Klares Berechnungsmodell
    - Problem asynchrone Ereignisse
      - z.B. Interrupt
- Ergebnisorientierte Terminierung
  - Prädikat über den Zustand des Gesamtsystems ist erfüllt
  - Achtung: Nicht jedes Prädikat geht



### Kommunikationsorientierte Terminierung

- Unterscheidung
  - Basisnachrichten
  - Kontrollnachrichten
- Prozesse sind aktiv oder passiv
- Actor-Modell
  - Kontrollnachrichten machen einen passiven Prozeß nicht aktiv
- Terminierung
  - Alle Prozesse sind passiv
  - Keine Basisnachrichten mehr unterwegs

Verteilte Systeme, Winter 2002
Folie 8.3

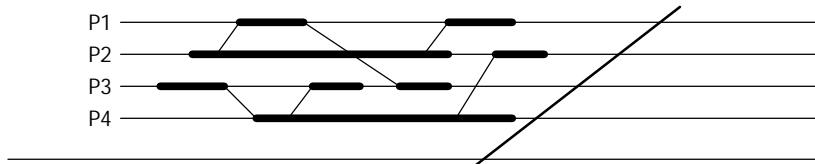
### Ein naiver Ansatz

Koordinator

- Koordinator sendet Multicast an beteiligten Prozesse
  - **Multicast(G, STATE)**
- Prozesse senden Unicast an Koordinator:
  - Zustand (aktiv/passiv)
  - Anzahl versendeter Nachrichten
  - Anzahl empfangener Nachrichten
  - **Unicast(Koor, state, m\_sent, m\_rcv)**
- Terminierung
 
$$\left( \forall_{p \in G} : state_p = passiv \right) \wedge \left( \sum_{p \in G} m\_sent_p = \sum_{p \in G} m\_rcv_p \right)$$

Verteilte Systeme, Winter 2002
Folie 8.4

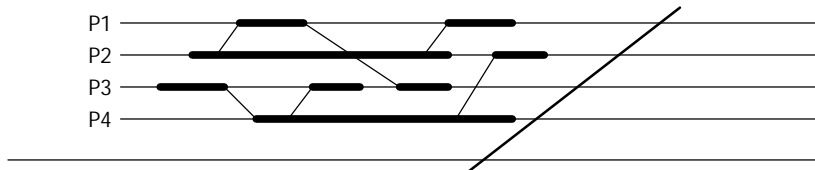
Funktioniert ... ?



Verteilte Systeme, Winter 2002

Folie 8.5

Funktioniert ... ?!



■ Terminiert?

- P1: Passiv, 2 empfangen und 1 gesendet
- P2: Passiv, 1 empfangen und 2 gesendet
- P3: Passiv, 2 empfangen und 1 gesendet
- P4: Passiv, 1 empfangen und 2 gesendet

■ Alle passiv und (6 empfangen = 6 gesendet)

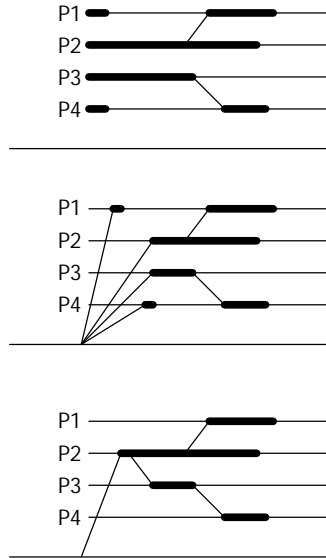
■ Terminiert!

Verteilte Systeme, Winter 2002

Folie 8.6

Am Anfang war ...

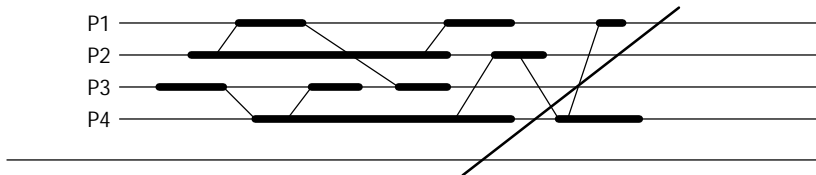
- Wie startet eine Berechnung im Actor-Modell
  - Beispiel war nicht korrekt
  - Prozesse werden „aus dem Nichts heraus“ aktiv
- Verschiedene Lösungen
- Jeder Prozeß beginnt aktiv
- „Gott“ sendet Geburtsnachricht an alle
  - Empfangszähler beginnt mit -1
- Startnachricht an einen Prozeß
  - Zählerstände?



Verteilte Systeme, Winter 2002

Folie 8.7

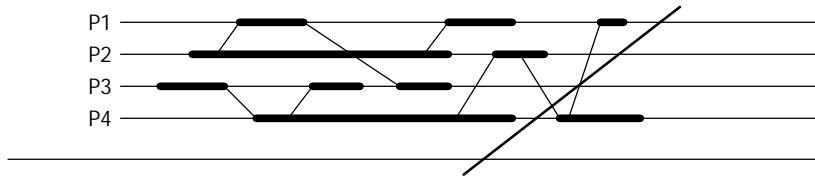
Funktioniert ... !?!



Verteilte Systeme, Winter 2002

Folie 8.8

Funktioniert ... Nicht !

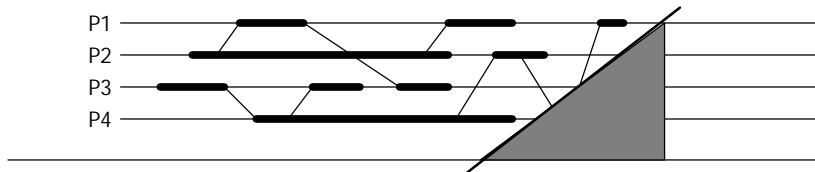


- Terminiert?
  - P1: Passiv, 3 empfangen und 1 gesendet
  - P2: Passiv, 1 empfangen und 3 gesendet
  - P3: Passiv, 2 empfangen und 1 gesendet
  - P4: Passiv, 1 empfangen und 2 gesendet
- Alle passiv und (7 empfangen = 7 gesendet)
- Terminiert!?

Verteilte Systeme, Winter 2002

Folie 8.9

Bermuda-Dreieck der verteilten Terminierung



- „Schiefe Bild“ des Koordinators inkonsistent
- Nachricht aus der Zukunft
- Korrekt, wenn Multicast zum selben Zeitpunkt bei allen Prozessen ankommt
  - Senkrechte Linie
- Gefährlich sind Nachrichten zwischen Anfang und Ende der Abfragewelle

Verteilte Systeme, Winter 2002

Folie 8.10

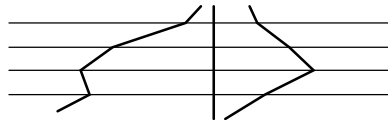
### Doppelzählverfahren

- Koordinator führt zwei Abfragen durch
  - Zweite Welle erst beginnen, wenn alle Antworten der ersten Welle beim Koordinator eingetroffen sind

- Terminierung

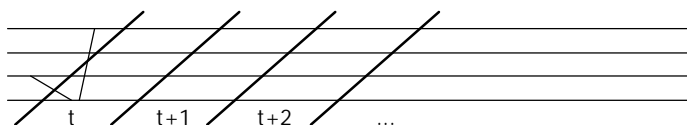
$$\left( \forall_{p \in G} : state_{p,1} = passiv \wedge state_{p,2} = passiv \right) \wedge \left( \sum_{p \in G} m\_sent_{p,1} = \sum_{p \in G} m\_recv_{p,1} = \sum_{p \in G} m\_sent_{p,2} = \sum_{p \in G} m\_recv_{p,2} \right)$$

- Anschaulicher Beweis



### Zeitzoneverfahren

- Basiert auf logischer Uhr (z.B. Lamportzeit)
- Realisierung
  - Jeder Prozeß besitzt logische Uhr
  - Jede Basisnachricht erhält Zeitstempel
  - Nur die Abfragen des Koordinators erhöhen Uhrzeit
- Prozesse erkennen Nachrichten aus der Zukunft
  - Prozeß setzt Flag und sorgt dafür, daß die nachfolgende Welle ungültig wird
- Was gewinnen wir gegenüber dem Doppelzählverfahren?



## Weitere Optimierungen

- Wann sendet ein Prozeß seinen Zustand?

Verteilte Systeme, Winter 2002

Folie 8.13

## Die Vektormethode

- Jeder Prozeß  $P_k$  besitzt lokalen Vektor  $V_k[n]$  (initial 0)
- $P_k$  sendet Basisnachricht an  $P_m$ :

$$VC_k[m] := VC_k[m] + 1;$$

- $P_k$  empfängt Basisnachricht von  $P_m$ :

$$VC_k[k] := VC_k[k] - 1;$$

- Kontrollvektor KV besucht alle Prozesse
- KV besucht  $P_k$ :

$$\begin{aligned} KV &:= KV + VC_k; \\ VC_k &:= (0, \dots, 0); \end{aligned}$$

$$VC_2 = \begin{pmatrix} +1 \\ -1 \\ +1 \\ +1 \end{pmatrix}$$

Verteilte Systeme, Winter 2002

Folie 8.14

Terminierungsbedingung

Prozeßsystem terminiert

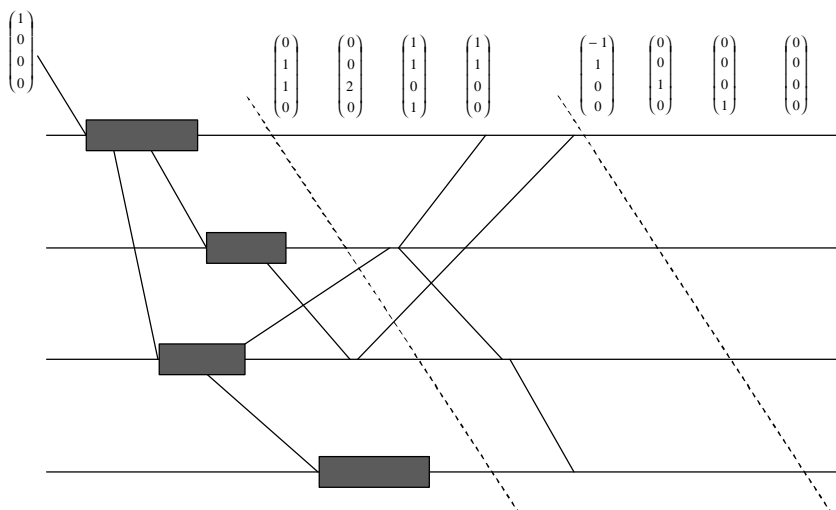


KV wird Nullvektor

Verteilte Systeme, Winter 2002

Folie 8.15

Ein Beispiel

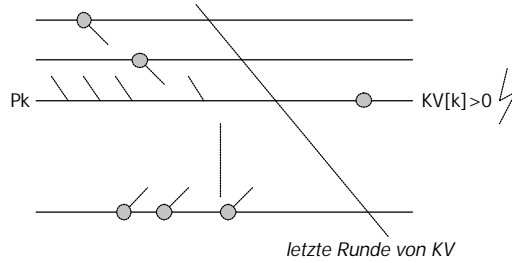


Verteilte Systeme, Winter 2002

Folie 8.16



### Korrektheit



- Invariante:

$$V_1[k] + \dots + V_k[k] + \dots + V_n[k] + KV[k]$$

Anzahl der unterwegs befindlichen Nachrichten für Prozeß k

- Behauptung: KV Nullvektor  $\Leftrightarrow$  Prozeßsystem terminiert
  - Prozeßsystem terminiert  $\Rightarrow$  KV Nullvektor
  - KV Nullvektor  $\Rightarrow$  Prozeßsystem terminiert
    - Widerspruchsbeweis: KV Nullvektor, aber keine Terminierung

### Nachlaufender Kontrollvektor

