Übung 3
Verteilte Systeme

Besprechung 13.1.2003

Matthias Ringwald,
Prof. Dr. R. Wattenhofer

RESEARCH GROUP FOR
Distributed Systems

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Aufgabe 1

- Gegeben: 3 Prozessoren, alle führen die gleiche Berechnung durch, verschiedene Systemarchitekturen:
  - Shared-Memory System
  - Asynchrones Message-Passing System
  - Semi-synchrones Message-Passing System
- Gesucht: Algorithmus für Mehrheitsentscheid
- Klar:
  - Bei 3 Prozessoren: Wenn 2 der gleichen Meinung sind, ist Mehrheit erreicht

Aufgabe 1

- Variante 1: Shared-Memory System, erlaubt atomaren Lese- oder Schreibzugriff
- Vorschlag Algo:
  - Jeder Prozeß schreibt sein Ergebnis in den Speicher
  - Danach liest er die beiden anderen Ergebnisse und entscheidet dann

Aufgabe 1

- Möglichkeiten:
  - Mindestens einmal das gleiche Ergebnis
    => fertig: Nimm eigenes Ergebnis
  - Beide haben ein anderes, gleiches Ergebnis
    => fertig. Nimm anderes Ergebnis
  - Kein oder anderes Ergebnis vorhanden
    => Warten?
- Warten, Möglichkeiten:
  - Warte konstante Zeit x Sekunden
    => Problem: Andere immer noch nicht fertig
  - Warte beliebig lange
    => Problem: Flugzeug stürzt ab.

Aufgabe 1

=> Frage: Gibt es eine Obergrenze für die Berechnung?

- nein: Gesamtsystem nicht funktionsfähig, unabhängig von der Systemarchitektur
- ja: eine konstante Wartezeit läßt sich bestimmen (und auch gleich Consensus machen, etc…)

=> Im weiteren Annahme einer Obergrenze

=> System funktionsfähig

Aufgabe 1

- Variante 2: Asynchrones Message-Passing System
- Unterschied zur Vorherigen Variante ?
  - Keine Garantie, dass Ergebnis den anderen übermittelt werden

=> System nicht funktionsfähig

Aufgabe 1

- Variante 3: Semi-synchrones Message-Passing System: Nachrichten kommen immer an und benötigen konstante Zeit
- Unterschied zu vorherigen Varianten ?
  - Falls die Nachrichtenlaufzeit berücksichtigt wird, ist die Variante gleich mächtig wie Variante 1

=> System funktionsfähig

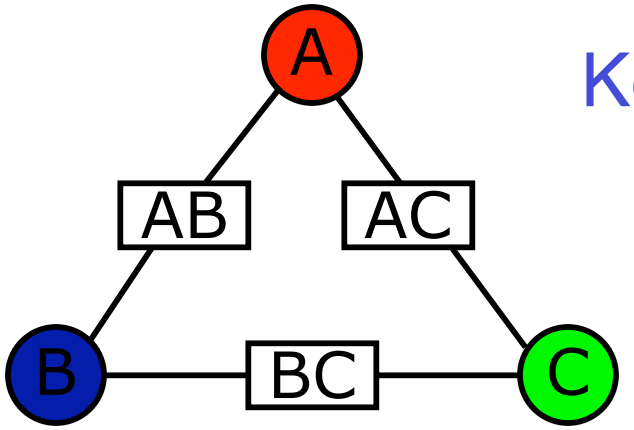
Aufgabe 2

- 1. Lösung: (kurz)
  - Wenn auf eine einzelne Speicherzelle nur von 2 Prozessoren zugegriffen werden kann, dann ist deren Consensus Nummer höchstens 2
  - Es gibt keine wait-free Implementierung, die aus Objekten mit Consensus Nummer 2 ein Objekt mit höherer Consensus Nummer konstruiert.
    (Siehe Theorem auf S. 95)
  - Die Consensus Nummer ist somit bei ‘non-trivial’ RMW Objekten 2

Aufgabe 2

- 2. Lösung (nach Schulbuch…)
- Möglicher Aufbau des Systems:

Eine Speicherzelle zwischen 2 Prozessoren



Kein Vorteil bei Verwendung mehrerer Speicherzellen

(x,y,z) ≈ Anfangszustand der 3 Prozessoren A,B,C

Aufgabe 2

- Mögliche Grundzustände:
  - 0-valenter Grundzustand
    (0,0,0) => 0
  - 1-valenter Grundzustand
    (1,1,1,) => 1
  - Es gibt einen bivalenten Grundzustand
    (0,1,x) => ?
- Wir betrachten diesen: (0,1,0)

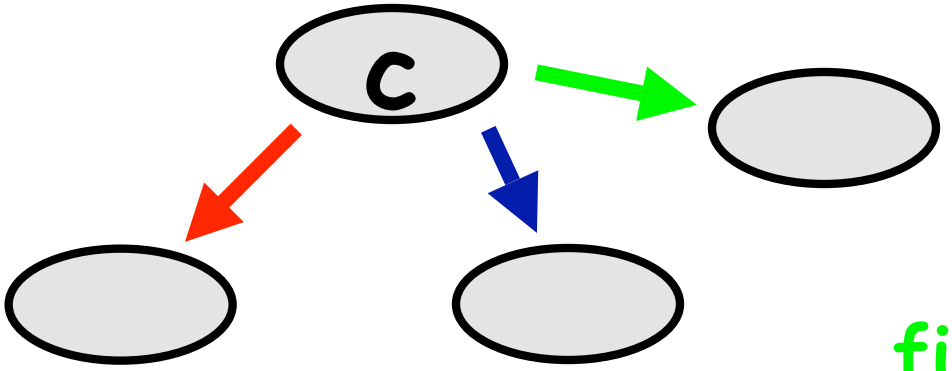
Aufgabe 2

- Kritischer Zustand
  - Es gibt mindestens einen kritischen Zustand
  - Falls beliebig oft von einem bivalenten Zustand in einen anderen bivalenten gewechselt werden kann, ist das System nicht wait-free, also muss ein kritischer Zustand erreicht werden

- Wir betrachten diesen kritischen Zustand und analysieren, welche Fälle dabei auftreten können

Aufgabe 2

From a Critical State



0-valent

If C goes first, protocol decides 0

0-valent

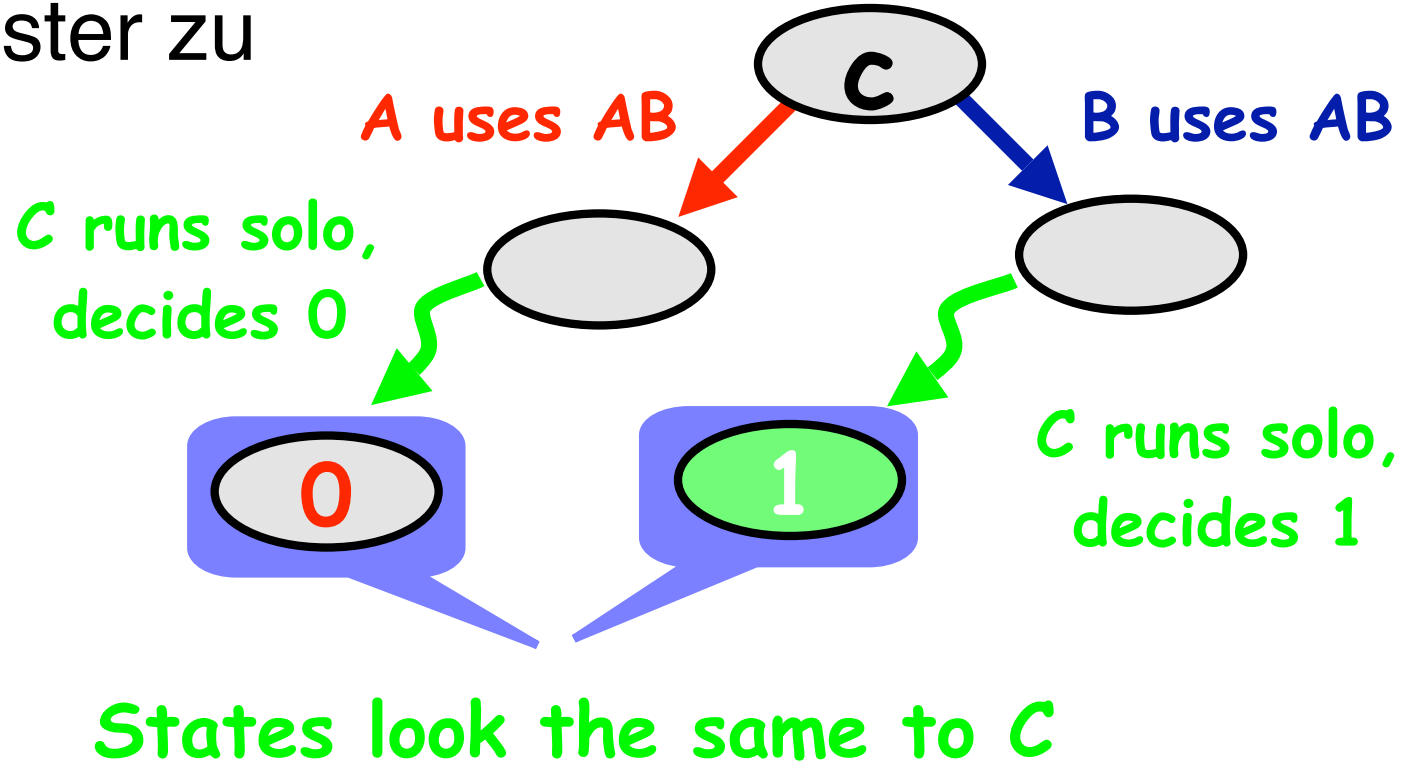
1-valent

If A goes first, protocol decides 0

If B goes first, protocol decides 1

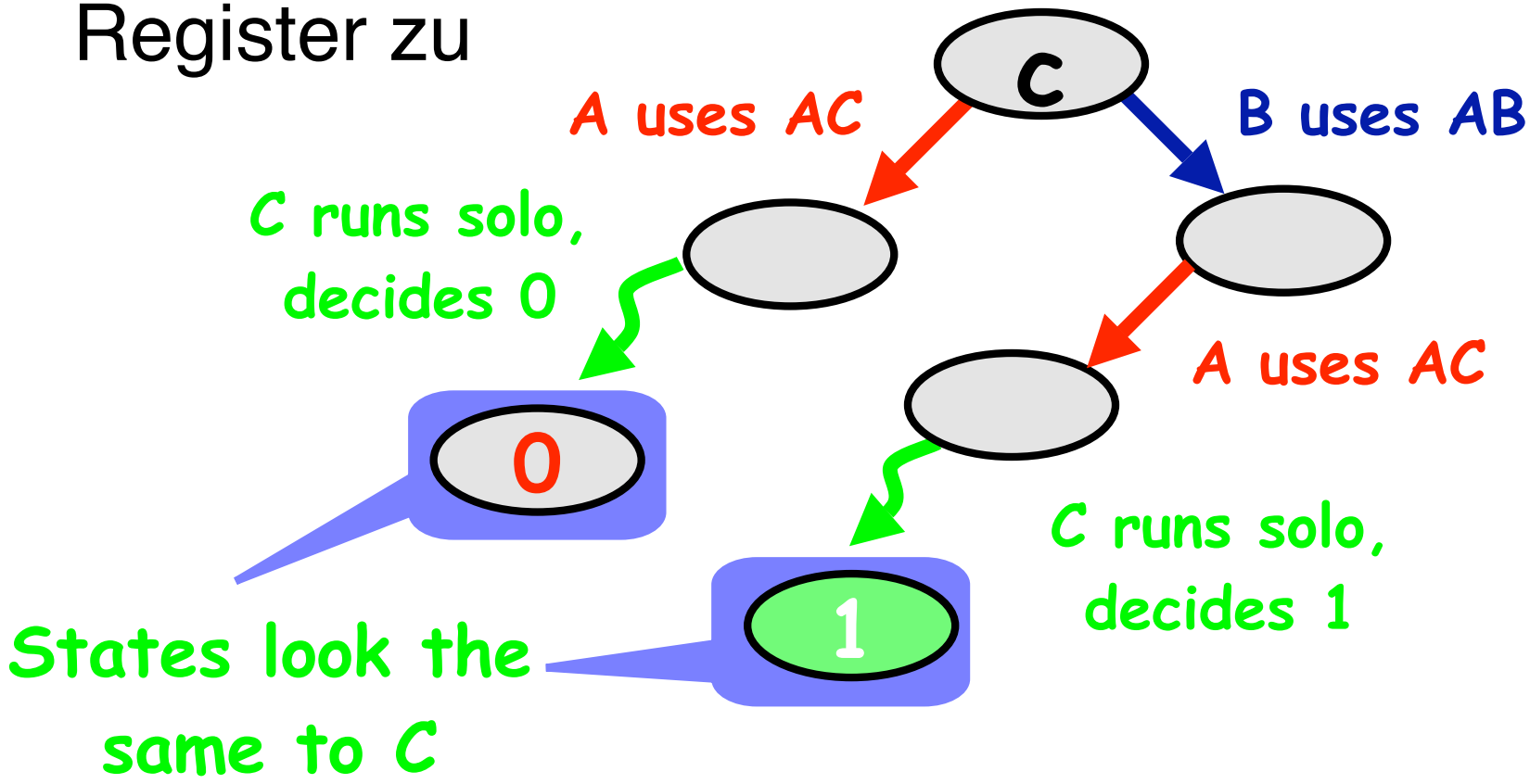
Aufgabe 2

a) A und B greifen auf das gleiche Register zu



Aufgabe 2

b) A und B greifen auf unterschiedliche Register zu



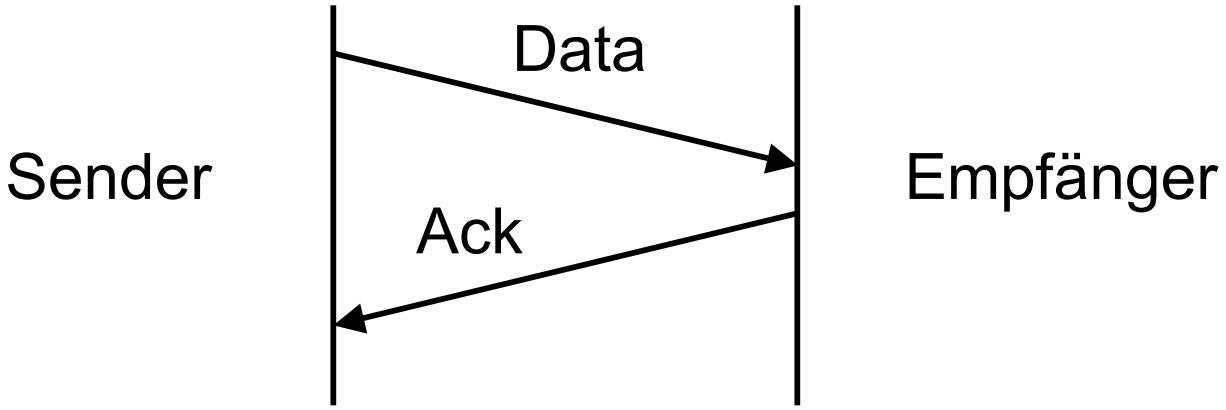
Aufgabe 2

- Ergebnis: Consensus von A,B und C nicht möglich
- Klar: Consensus zwischen 2 Prozessoren, z.B. A und B ist über deren gemeinsame Speicherzelle möglich

=> Consensus Nummer = 2

Aufgabe 3

- Nein.



Selbst wenn bei TCP/IP der Sender die Bestätigung über das Ausliefern der Daten erhalten hat, weiß der Empfänger nicht, ob der Sender diese Bestätigung auch bekommen hat.