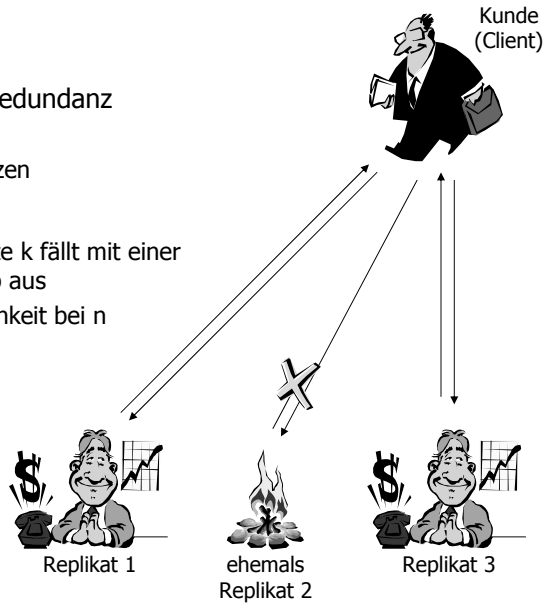
Verteilte Systeme

13. Fehlertoleranz

Motivation

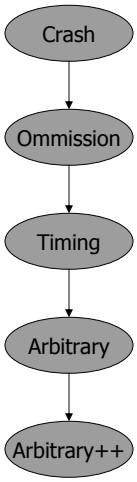
- Verteiltes System = Redundanz
  - Rechnern
  - Kommunikationsnetzen
- Grundidee
  - Einzelne Komponente k fällt mit einer Wahrscheinlichkeit p aus
  - Ausfallwahrscheinlichkeit bei n Replikaten von k:

$$p^n < p$$



Verteilte Systeme, Sommersemester 1999 Folie 13.2

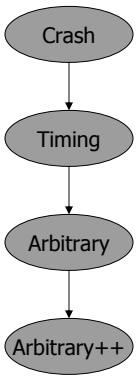
Fehlermodelle



- Welche Fehler soll eine Anwendung tolerieren?
  - Maskieren von Fehlern
- Aufeinander aufbauende Fehlerklassen
  - *Crash*
    - Sofortiger, dauerhafter und beobachtbarer Stop (Fail-and-Stop oder auch Fail-Silent)
    - Grad der Amnesie
  - *Ommission*
    - Auslassung
  - *Timing*
    - Zu früh (Echtzeitfehler)
    - Zu spät (Performance-Fehler)
  - *Arbitrary*
    - Beliebige Fehler, aber ehrlich gemeint
  - *Arbitrary with Message Authentication*
    - Beliebige Fehler inklusive absichtlicher Fälschungen

Verteilte Systeme, Sommersemester 1999 Folie 13.3

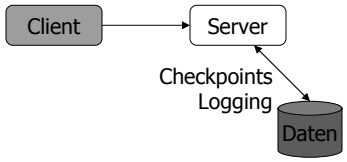
Modellvariante

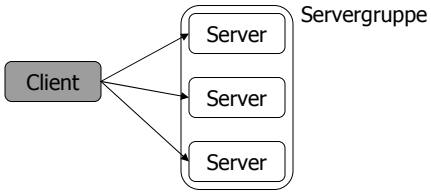


- Ommission wird durch Wiederholungen beliebig unwahrscheinlich
- Übergang zu Timing-Fehlern
- Timing-Fehler sind zeitlich begrenzt
- Entdeckung in endlicher Zeit (Timeout)

Verteilte Systeme, Sommersemester 1999 Folie 13.4

Ansätze

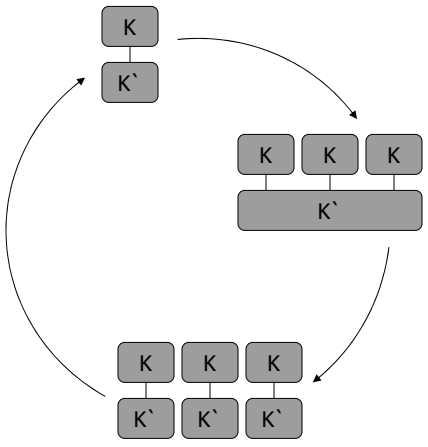




- Passive Redundanz
- Replikation von
  - Daten
- Vorteil: Einfache Realisierung
- Nachteil: Nicht alle Fehlerklassen maskierbar
  - Welche nicht?

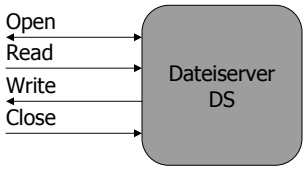
- Aktive Redundanz
- Replikation von
  - Daten
  - Kontrollfluß
- Vorteil: Alle Fehlerklassen im Prinzip maskierbar
- Nachteil: Aufwendige Hardware- und Software-Realisierung

Mathematischer Hintergrund



- Einzelne Komponente hat Ausfallwahrscheinlichkeit p
- Ziel: N-fache Replikation der mit Ausfallwahrscheinlichkeit p^n
- Ausfälle müssen unabhängige Ereignisse sein
  - Andere Komponenten
  - Systemsoftware
  - Hardware
- Zuverlässigkeit der Komponente hängt von vielen anderen Komponenten im System ab
- Gesamtsystem kann nicht zuverlässiger als die unzuverlässigste Komponente werden

Beispiel



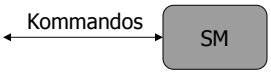
- Einfacher, nicht fehlertoleranter Dateiserver DS
- Maskierung maximal eines Ausfalls
- Realisierung
  - Crash?
  - Ommission?
  - Timing?
  - Arbitrary?
  - Arbitrary mit Nachrichtenauthentifizierung?

Wieviel Redundanz ist notwendig

- Abhängig von
  - Maskierte Fehlerklasse
  - Tolerierbare Anzahl an gleichzeitigen Ausfällen
- K-Zuverlässigkeit
  - Servergruppe toleriert den gleichzeitigen Ausfall von k Mitgliedern
- Replikationsgrad k+1
  - Crash, Ommission und Timing
  - Schnellste gewinnt
- Replikationsgrad 2k+1
  - Arbitrary
  - Mehrheitsentscheid
- Replikationsgrad 3k+1
  - Arbitrary with Message Authentication

Aktive Redundanz: State-Machine-Approach

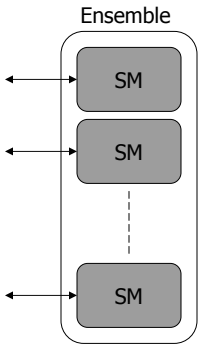
- Zustandsmaschine
  - Zustandsvariablen
  - Zustandsverändernde Kommandos: `a=sm.k(e)`
- Determinismus und Atomarität
  - a hängt nur vom initialen Zustand und der Abarbeitungsreihenfolge vorangegangener Kommandos ab
- Funktion der Zustandsmaschine unabhängig von
  - Zeit
  - Anderen Systemaktivitäten
- Keine Seiteneffekte
- Sind Zustandsmaschinen leicht realisierbar?



Verteilte Systeme, Sommersemester 1999 Folie 13.9

Ein Ensemble

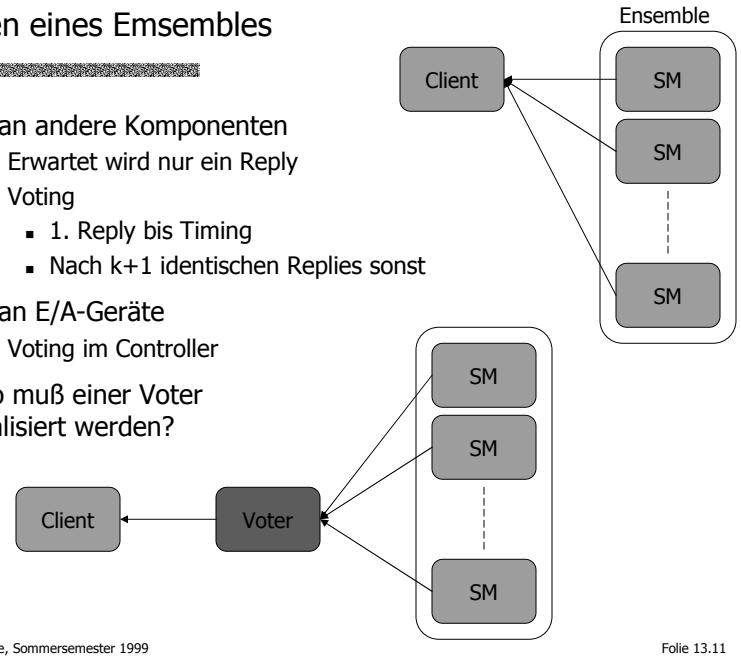
- Replizierte State-Machines
- Aktive Redundanz bei Gleichtaktung
  - Gleicher initialer Zustand
  - Kommandos in der gleichen Reihenfolge
- Replikatkoordination
  - Alle nicht-fehlerhaften Zustandsmaschinen erhalten alle Kommandos
  - Ausführung der Kommandos in der gleichen Reihenfolge
- Realisierungsvarianten
  - Eigene Implementierung
  - Geordneter Multicast und ...?
    - Welcher Ordnungsgrad ist minimal notwendig?



Verteilte Systeme, Sommersemester 1999 Folie 13.10

Ausgaben eines Emsembles

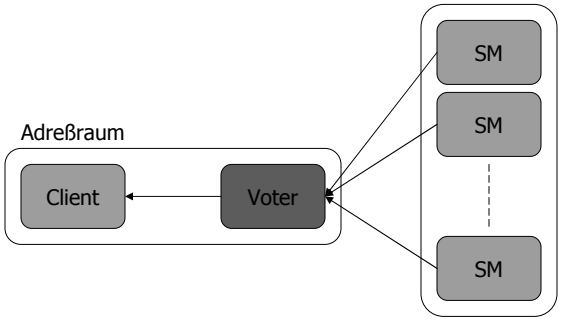
- ... an andere Komponenten
  - Erwartet wird nur ein Reply
  - Voting
    - 1. Reply bis Timing
    - Nach k+1 identischen Replies sonst
- ... an E/A-Geräte
  - Voting im Controller
- Wo muß einer Voter realisiert werden?



Verteilte Systeme, Sommersemester 1999 Folie 13.11

Lokalisierung Voter

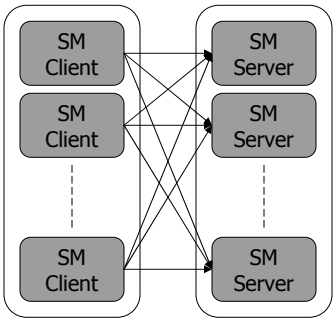
- Warum nicht als eigenständiger Prozeß?
- Voter ist Teil einer Laufzeitbibliothek im Client



Verteilte Systeme, Sommersemester 1999 Folie 13.12

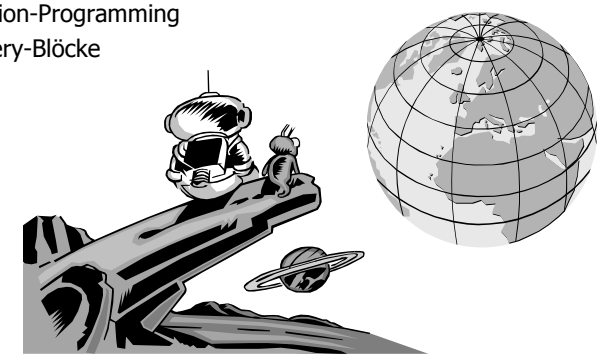
Unzuverlässige Clients

- Fehlerhafte Clients können k-zuverlässiges Ensemble stören
  - Trotzdem alle Aufträge in gleicher Reihenfolge bearbeitet werden
  - Vergleichbar "Denial of Service"-Attacken
- K-zuverlässiges Ensemble $\not\Rightarrow$ k-zuverlässiges System
- Lösungsansätze
  - Defensive Programmierung
  - Clients ebenfalls replizieren
    - Zu maskierende Fehlerklasse?
    - Plazierung des serverseitigen Voters?



Wirklich Fehlerklasse "Arbitrary"?

- Maskierung von Softwarefehlern
- Einfache Replizierung einer State-Machine reicht nicht
- Lösungsansätze
  - N-Version-Programming
  - Recovery-Blöcke
- Aufwand



Rekonfigurierung

- Reaktion auf ausgefallene Replikate
- Fehlerklasse Crash bis Timing
  - Erzeugung eines Ersatzservers
  - Einphasen mit dem Ensemble
    - Selbststabilisierende Replikate (vgl. Dijkstra 1974)
    - Abgleich über korrektes Ensemblemitglied
- Fehlerklasse Arbitrary
  - Kein Ersatz integrierbar
  - Zuverlässigkeitsgrad wird kleiner
  - Aufwendige und kontrollierte Neuinitialisierung bei zu geringem Zuverlässigkeitsgrad

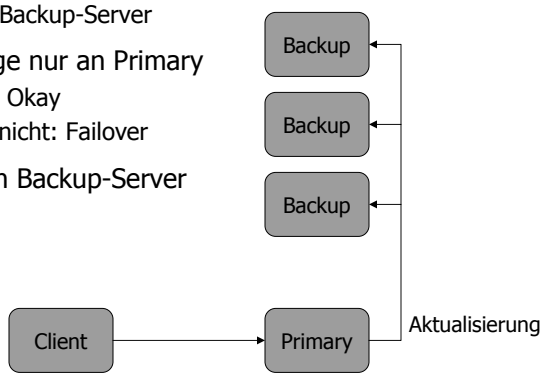
Zusammenfassung State-Machine-Approach

- Ein Ansatz für aktive Redundanz
  - Einschränkungen an die Replikat-Realisierung
    - Determinismus
    - Atomarität
  - Total geordneter Multicast
  - Anwendungsunabhängige Voter
    - Mehrheitsentscheid: Bitweiser Vergleich der Replies
- Materialschlacht
- Zuverlässigkeitsgrad k>1 selten

Aktive Redundanz selbst gestrickt?

Passive Redundanz: Primary-Backup-Approach

- Ein Primary-Server
- Backup-Server im "Hintergrund"
  - Aktualisierung der Backup-Server
- Client sendet Aufträge nur an Primary
  - Primary antwortet: Okay
  - Primary antwortet nicht: Failover
- Im Fehlerfall wird ein Backup-Server zum neuen Primary



Aktualisierungsfrequenz

- Hot Standby
  - Jedes Kommando wird nachgezogen
- Warm Standby
  - Periodische Übernahme des Primaryzustands
- Cold Standby
  - Übernahme des Primaryzustands nach Ausfall
- Unterschiede in der Reaktionszeit
- Election bei Primaryausfall