Verteilte Systeme

7. Fehlertoleranz

Sommersemester 2011

Institut für Betriebssysteme und
Rechnerverbund

TU Braunschweig

Dr. Christian Werner
– Bundesamt für Strahlenschutz –

INSTITUT FÜR BETRIEBSSYSTEME
UND RECHNERVERBUND
Prof. Dr.-Ing. L. Wolf | Prof. Dr. S. Fekete

IBR

Überblick

- Motivation für Fehlertoleranz in VS
- Fehlermodelle
- Erreichen von Fehlertoleranz
  - Ausfallsicherheit von Prozessen
  - Zuverlässiger Remote Procedure Call

Fehlertoleranz

- Charakteristische Eigenschaft verteilter Systeme: partielle Fehler/Ausfälle (anders als Ein-Maschinen-Systeme)
- Durch partielle Fehler können Komponenten beeinflusst werden, während andere problemlos weiter arbeiten können
- Daher Ziel in vert. Systemen: konstruiere System so, dass es partielle Ausfälle verkraften kann, ohne dass die Leistungsfähigkeit zu sehr leidet
- Insbesondere sollte es während der Reparatur weiter arbeiten
- Das System ist dann fehlertolerant (fault-tolerant)

Verlässliche Systeme

- Fehlertoleranz hängt eng mit verlässlichen Systemen zusammen.
- Verlässlichkeit umfasst mehrere Konzepte:
  - Verfügbarkeit: das System ist sofort benutzbar
  - Zuverlässigkeit: das System läuft fortwährend ohne Fehler
  - Sicherheit (Safety): „nothing bad happens“
  - Wartbarkeit: sagt aus, wie leicht und schnell ein ausgefallenes System repariert werden kann
- Die Fähigkeit, verlässliche Systeme zu bauen, hängt stark von der Fähigkeit ab, auf Ausfälle reagieren zu können.

Fehlermodelle

- Zweck:
  - Klassifikation von Fehlern
  - Angabe der Fehlertoleranz von Programmen mit Bezug auf die Fehlerklassen

Type of failure	**Description**
Crash failure	A server halts, but is working correctly until it halts
Omission failure *Receive omission* *Send omission*	A server fails to respond to incoming requests A server fails to receive incoming messages A server fails to send messages
Timing failure	A server's response lies outside the specified time interval
Response failure *Value failure* *State transition failure*	The server's response is incorrect The value of the response is wrong The server deviates from the correct flow of control
Arbitrary failure	A server may produce arbitrary responses at arbitrary times

Maskierung von Fehlern

- Beste Vorgehensweise: Verbergen der Fehler vor anderen Komponenten (Maskierung)
- Wird erreicht durch *Redundanz*
  - Informationsredundanz:
    - Verwendung zusätzlicher Bits, um den möglichen Ausfall anderer Bits abzufangen
    - Beispiele: Forward Error Correction, Audio-CD
  - Zeitliche Redundanz:
    - Wiederholung von Aktionen
    - Beispiel: Transaktionen
  - Physische Redundanz
    - Zusätzliche Hardware oder Prozesse

Maßnahmen zur Steigerung der Fehlertoleranz

- Wie kann man Fehlertoleranz erreichen?
- Basismechanismus: Replikation, daher enger Zusammenhang zu Fehlertoleranz!

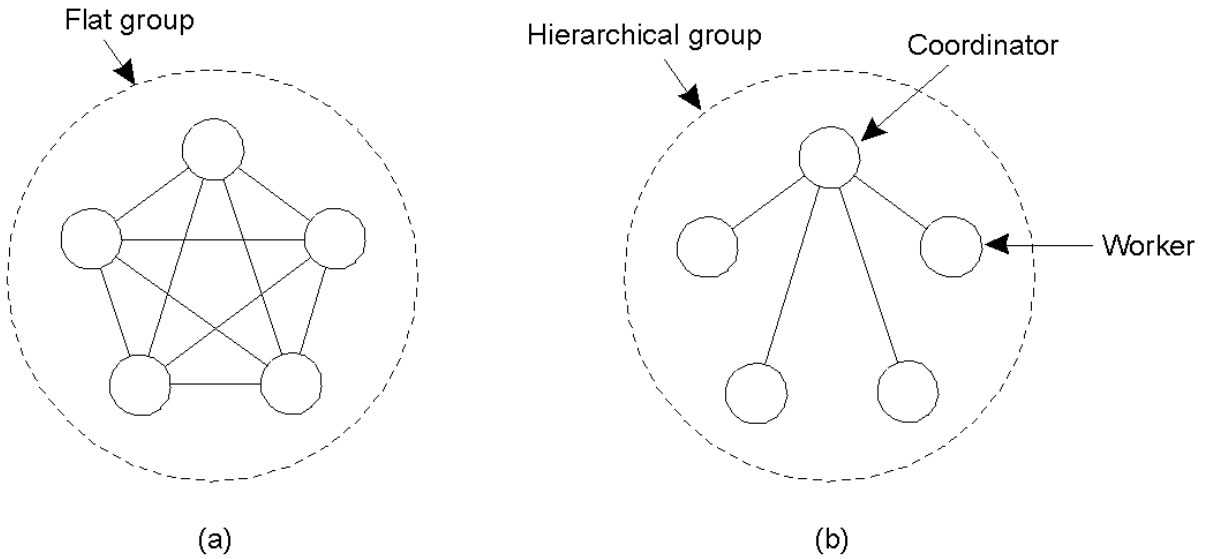
Prozess-Ausfallsicherheit

- Generelles Vorgehen:
  - Replikation von Prozessen (die dann die gleiche Aufgabe erfüllen)
  - Zusammenfassung in Prozessgruppen
  - Ergebnis: fehlertolerante Gruppe von Prozessen
  - Wird eine Nachricht an die Gruppe geschickt, bekommt jeder Prozess die Nachricht
- Wir betrachten
  - Organisation der Gruppen
  - Fehlermaskierung und Replikation
  - Erzielung von Übereinstimmung in fehlerhaften Systemen

Gruppenorganisation

- Dynamische Gruppen
  - Können erzeugt und gelöscht werden
  - Prozesse können ein- und austreten
  - Prozesse können Mitglied mehrerer Gruppen sein
- Zweck des Einsatzes von Gruppen: Abstraktion von den Einzelprozessen, Darstellung gegenüber der Außenwelt als eine Einheit
- Gruppenorganisation:
  - strukturlos oder
  - Hierarchisch (ein Koordinator)

Gruppenorganisation



- Kein single-point-of-failure
- Jede Entscheidung erfordert Abstimmung

- Schneller im Normalbetrieb
- Problem bei Ausfall des Koordinators

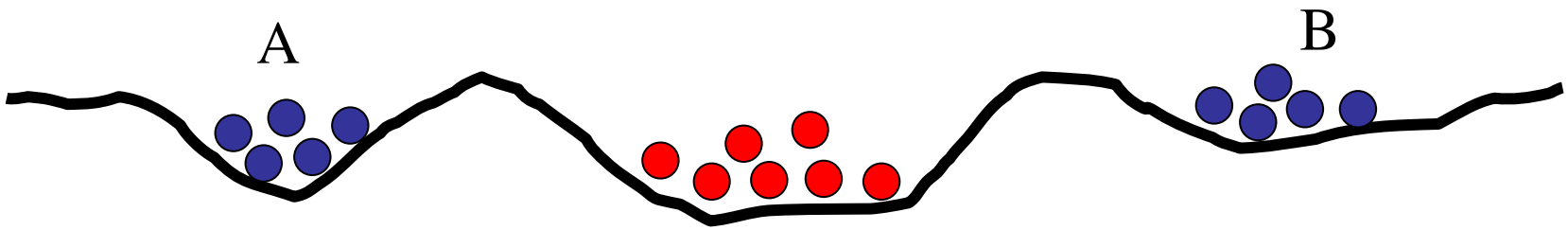
Maskierung von Fehlern

- Anwendung der Replikationsstrategien
  - Primary-Based:
    - Koordinator, der alle Write-Operationen koordiniert
    - Wenn der Koordinator abstürzt, wählen die übrigen Prozesse einen neuen
  - Active Replication:
    - Passt gut zu flachen Gruppen
- Wichtige Frage: wieviel Replikation ist nötig?
  - Hängt stark vom Fehlerverhalten ab
  - Wenn Prozesse byzantinische Fehler zeigen, dann benötigt man 2k+1 Prozesse, um k Fehler abfangen zu können
  - Kann man aber so genau sagen, dass maximal k Fehler auftreten? -> statistische Analysen

Übereinstimmung in fehlerhaften Systemen

- Problem: wie kommt man zu einer Übereinstimmung über eine durchzuführende Aktion im Fall
  - Perfekte Prozesse, aber fehlerhafte Kommunikationskanäle
  - Fehlerhafte Prozesse, aber perfekte Kommunikation
- Analogien:
  - Das 2-Armeen-Problem
  - Das Problem der byzantinischen Generäle

Das 2-Armeen-Problem

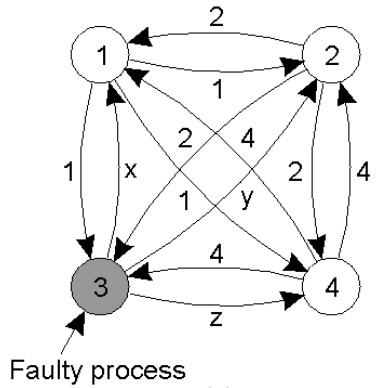


- Blau will rot angreifen, kann aber nur gemeinsam gewinnen (Überzahl)
- Notwendig: Abstimmung über Zeitpunkt des Angriffs
- Bote muss durch das rote Lager, kann gefangen werden -> unzuverlässige Übertragung
- Welches Problem hat B, wenn der Bote sicher bei A angekommen ist?
- Man kann zeigen, dass die beiden Prozesse nicht zu einer Übereinstimmung kommen können.

Die byzantinischen Generäle

- Szenario: rote Armee im Tal, n blaue Armeeteile in den Hügeln
- Kommunikation über zuverlässige Verbindung (Telefon)
- Problem: m der n Generäle sind Verräter, versuchen die Übereinstimmung der anderen zu verhindern
- Frage: können die loyalen Generäle trotzdem eine Übereinstimmung erzielen?
- Unser Freund Lamport hat auch hierzu eine Lösung ☺. Funktioniert, wenn mehr als 2/3 aller Generäle loyal sind.
- Beispiel: Übereinstimmung über die Truppenzahl soll erreicht werden

Lösung für die Generäle



(a)

1 Got(1, 2, x, 4)
2 Got(1, 2, y, 4)
3 Got(1, 2, 3, 4)
4 Got(1, 2, z, 4)

(b)

1 Got	2 Got	4 Got
(1, 2, y, 4)	(1, 2, x, 4)	(1, 2, x, 4)
(a, b, c, d)	(e, f, g, h)	(1, 2, y, 4)
(1, 2, z, 4)	(1, 2, z, 4)	(i, j, k, l)

(c)

- Szenario: n=4, m=1
- G1, G2 und G4 nennen die korrekte Stärke, G3 lügt zu allen
- 4 Schritte
- Schritt 1 (Bild (a)): alle Generäle melden ihre Stärke an alle anderen →in Schritt 2 Ergebnis in Vektor Bild (b)
- Schritt 3: jeder General meldet seinen Vektor aus (b) an alle anderen (G3 lügt wieder heftig) → Ergebnis in (c)
- Schritt 4: Jeder General überprüft die Spalten, wenn es eine Mehrheit gibt, ist der Wert korrekt
- Übereinstimmung: (1,2,unknown, 4)

Zusammenfassung

- Fehlertoleranz ist ein wichtiges Konzept in verteilten Systemen
  - Es gibt viel mehr Fehlerquellen als in zentralisierten Systemen
  - Das System funktioniert möglicherweise trotz teilweiser Fehler.
- Wir haben exemplrisch Lösungen betrachtet für die Koordination von redundanten Prozessen.

Diskussion