Verteilte Systeme

Vorlesung 10
Sebastian Iwanowski
FH Wedel

Client-basierte Konsistenz

Monotones Lesen:

Client bekommt beim wiederholten Lesen desselben Datensatzes mindestens genau so aktuelle Werte zurück wie beim letzten Mal

Read your Writes:

Client bekommt beim wiederholten Lesen desselben Datensatzes genau dieselben Werte wie die, die er schon einmal geschrieben hat.

Monotones Schreiben:

Bevor ein Client einen neuen Wert in eine Kopie schreibt, muss er einen älteren Wert, den er vorher schon in andere Kopien geschrieben hat, auch in diese Kopie schreiben.

Write follows Read:

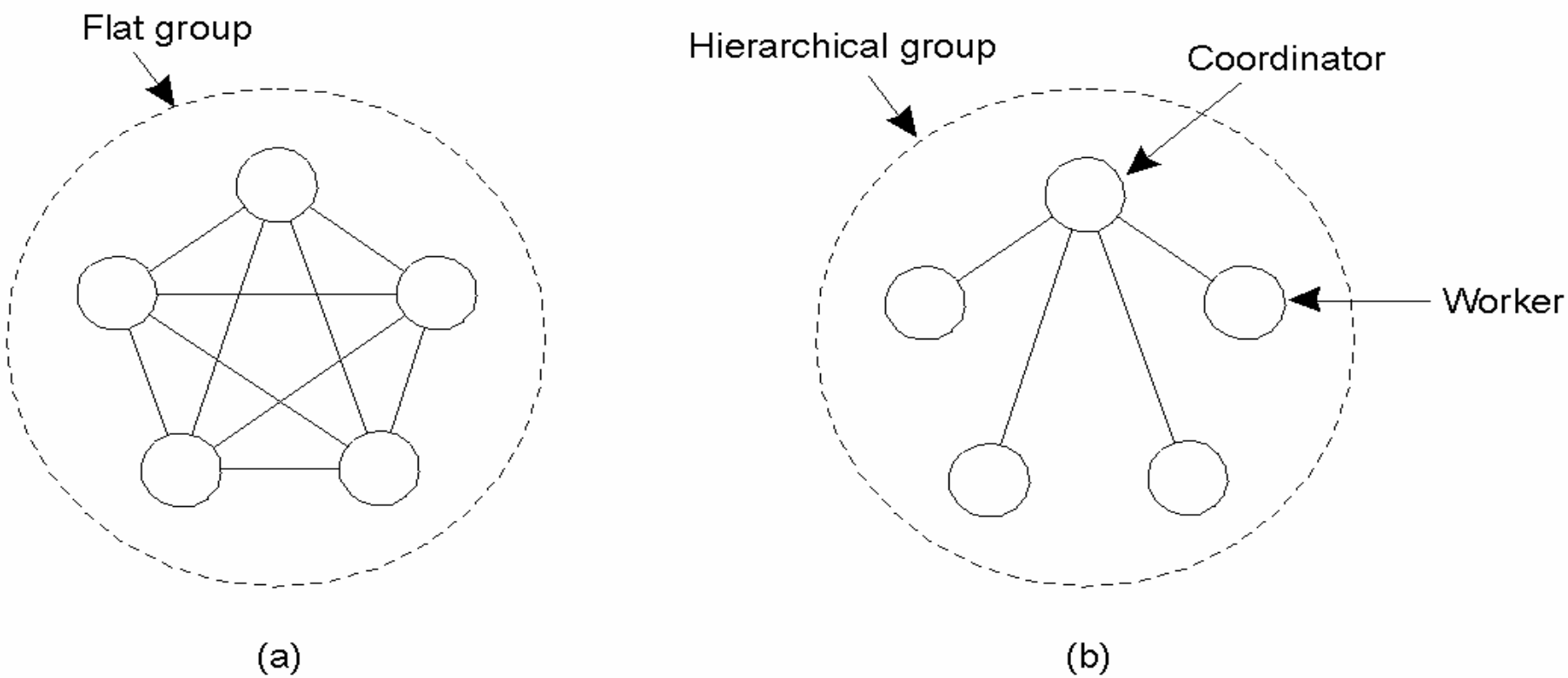
Bevor ein Client einen neuen Wert in eine Kopie schreibt, muss ein älterer Wert, den er vorher bereits von einer anderen Kopie gelesen hat, auch in dieser Kopie stehen.

Verteilte Systeme

1. Innovative Beispiele aus der Praxis
2. Allgemeine Anforderungen und Techniken verteilter Systeme
3. Die Client-Server-Beziehung und daraus entstehende Fragestellungen
4. Nebenläufigkeitstechniken in Java
5. Entfernte Aufrufe
6. Objektmigration
7. Agententechnologie
8. Dienstevermittlung
9. Synchronisation von Daten
10. Konzepte zur Erzielung von Fehlertoleranz
11. Web Services

Koordination der redundanten Server

Flache vs. hierarchische Koordination:



Fehlermaskierungsverfahren (Server)

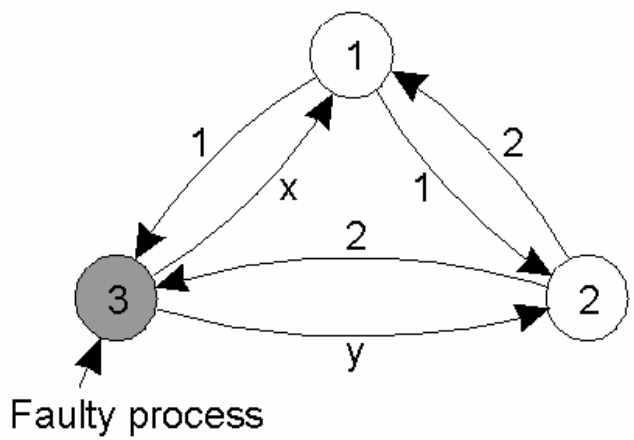
Lösungsverfahren für die flache Koordination: Mehrheitsentscheid

a) **Die Server tauschen sich paarweise aus.**

b) **Jeder Server bildet Vektoren aus den Werten, die er von den anderen Servern erhalten hat.**

c) **Die Server tauschen paarweise diese Vektoren aus.**

d) **Als gültige Werte werden dann die Mehrheiten an den jeweiligen Positionen angenommen**



(a)

Beispiel mit 3 Servern:

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

(b)

1 Got	2 Got
(1, 2, y)	(1, 2, x)
(a, b, c)	(d, e, f)

(c)

Fehlermaskierungsverfahren (Server)

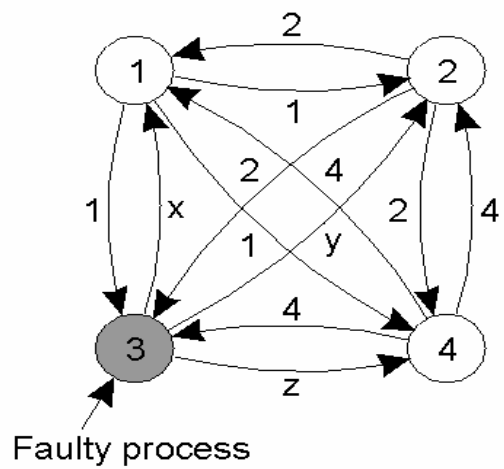
Lösungsverfahren für die flache Koordination: Mehrheitsentscheid

a) **Die Server tauschen sich paarweise aus.**

b) **Jeder Server bildet Vektoren aus den Werten, die er von den anderen Servern erhalten hat.**

c) **Die Server tauschen paarweise diese Vektoren aus.**

d) **Als gültige Werte werden dann die Mehrheiten an den jeweiligen Positionen angenommen**



(a)

Beispiel mit 4 Servern:

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)

Fehlermaskierungsverfahren (Server)

3 anscheinend verschiedene Problemstellungen:

1) **Übereinstimmung (Consensus):**
   Vor.: Alle Server erhalten gleiche Werte.
   Ziel: Die korrekten Server haben den richtigen Wert.

2) **Byzantinische Generäle:**
   Vor.: Alle Server erhalten gleichen Wert von Koordinator.
   Ziel: Die korrekten Server haben den richtigen Wert.

3) **Interaktive Übereinstimmung:**
   Vor.: Alle Server erhalten einen beliebigen Wert.
   Ziel: Die korrekten Server haben die Werte aller anderen korrekten Server.

Satz: Die Probleme sind äquivalent

Beweis: siehe Coulouris, S. 454 (englisch) / S. 526 (deutsch)

Fehlermaskierungsverfahren (Server)

Satz von Lamport (1982):

Im Problem der Byzantinischen Generäle braucht man mindestens 3f+1 Server, um f Fehler maskieren zu können.

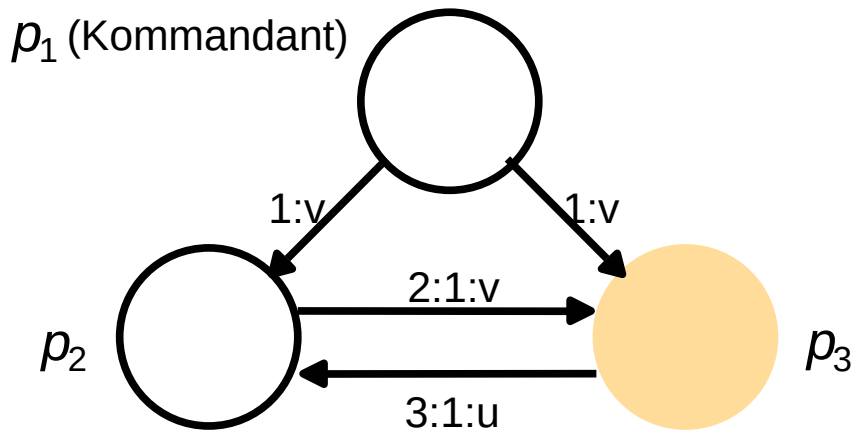
Beweisskizze: **siehe Coulouris, S. 457 (englisch) / S. 530 (deutsch)**

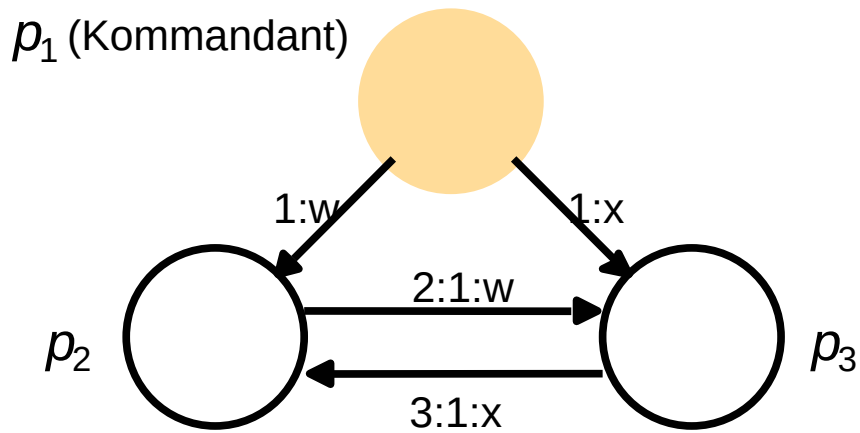
Fehlermaskierungsverfahren (Server)

Satz von Lamport (1982):

Im Problem der Byzantinischen Generäle braucht man mindestens 3f+1 Server, um f Fehler maskieren zu können.

Beispiel für f = 1: **3 Server reichen nicht aus !**





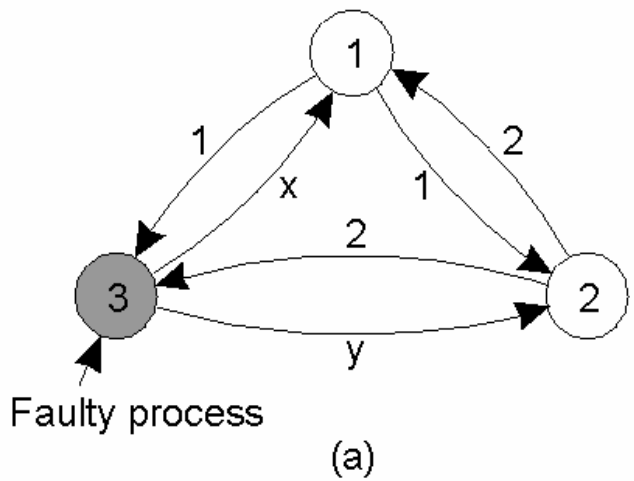
Fehlerhafte Server sind schattiert

Fehlermaskierungsverfahren (Server)

Satz von Lamport (1982):

Im Problem der Byzantinischen Generäle braucht man mindestens 3f+1 Server, um f Fehler maskieren zu können.

Beispiel für f = 1: **Problem *Interaktive Übereinstimmung,***
Beispiel in Tanenbaum, S. 423



(a)

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

(b)

1 Got	2 Got
(1, 2, y)	(1, 2, x)
(a, b, c)	(d, e, f)

(c)

Gegenbeispiel ?

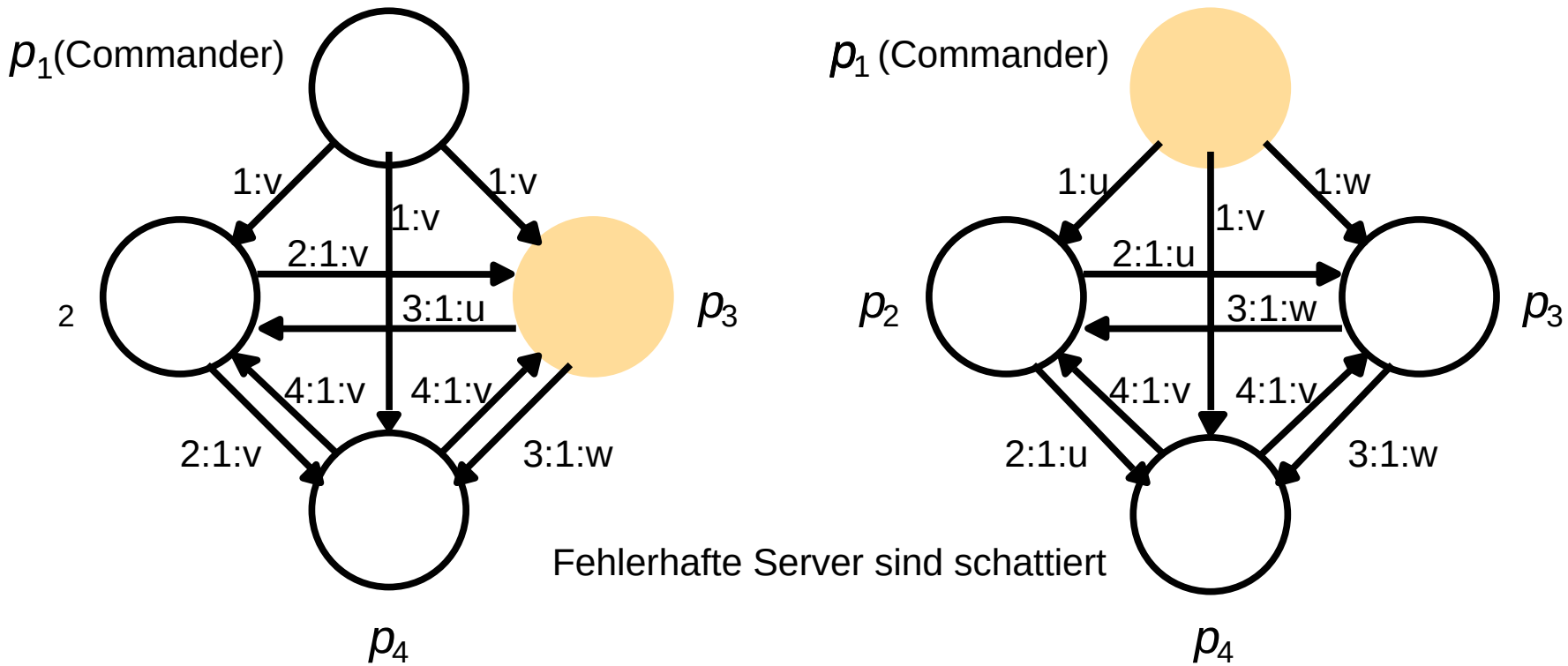
Was hat Tanenbaum nicht bedacht ?

Fehlermaskierungsverfahren (Server)

Satz von Lamport (1982):

Im Problem der Byzantinischen Generäle braucht man mindestens 3f+1 Server, um f Fehler maskieren zu können.

Beispiel für f = 1: **4 Server reichen aus !** **Wirklich ?**

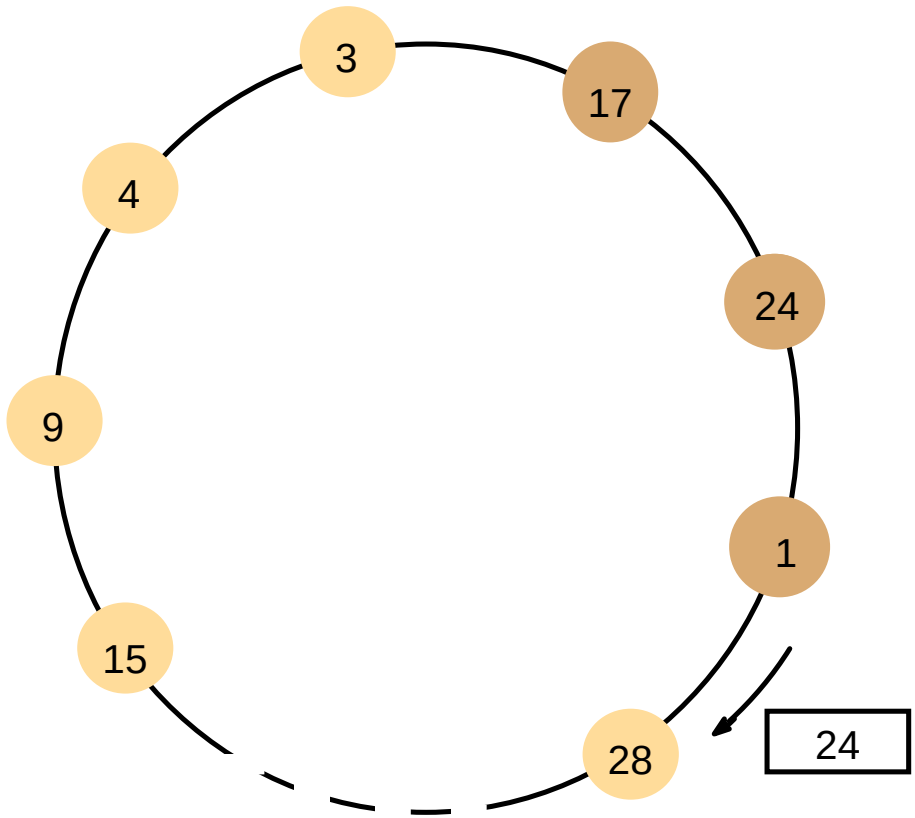


Fehlerhafte Server sind schattiert

Verteilte Auswahl eines Koordinators

Verfahren für ringförmig verbundene Server (Chang / Roberts 1979):

1) **Am Anfang ist jeder Server nichtmarkiert.**
2) **Irgendein Server beginnt, markiert sich und sendet seine ID dem Nachbarn.**
3) **Jeder Server, der eine ID empfängt, vergleicht diese mit der eigenen:**
   a) **Fremde ID > eigene ID => Server markiert sich und sendet fremde ID weiter.**
   b) **(Fremde ID < eigene ID) und (Server noch nicht markiert) => Server markiert sich und sendet fremde ID weiter.**
   c) **(Fremde ID < eigene ID) und (Server bereits markiert) => nichts**
   d) **Fremde ID = eigene ID => Dieser Server wird der Koordinator !**



Verteilte Auswahl eines Koordinators

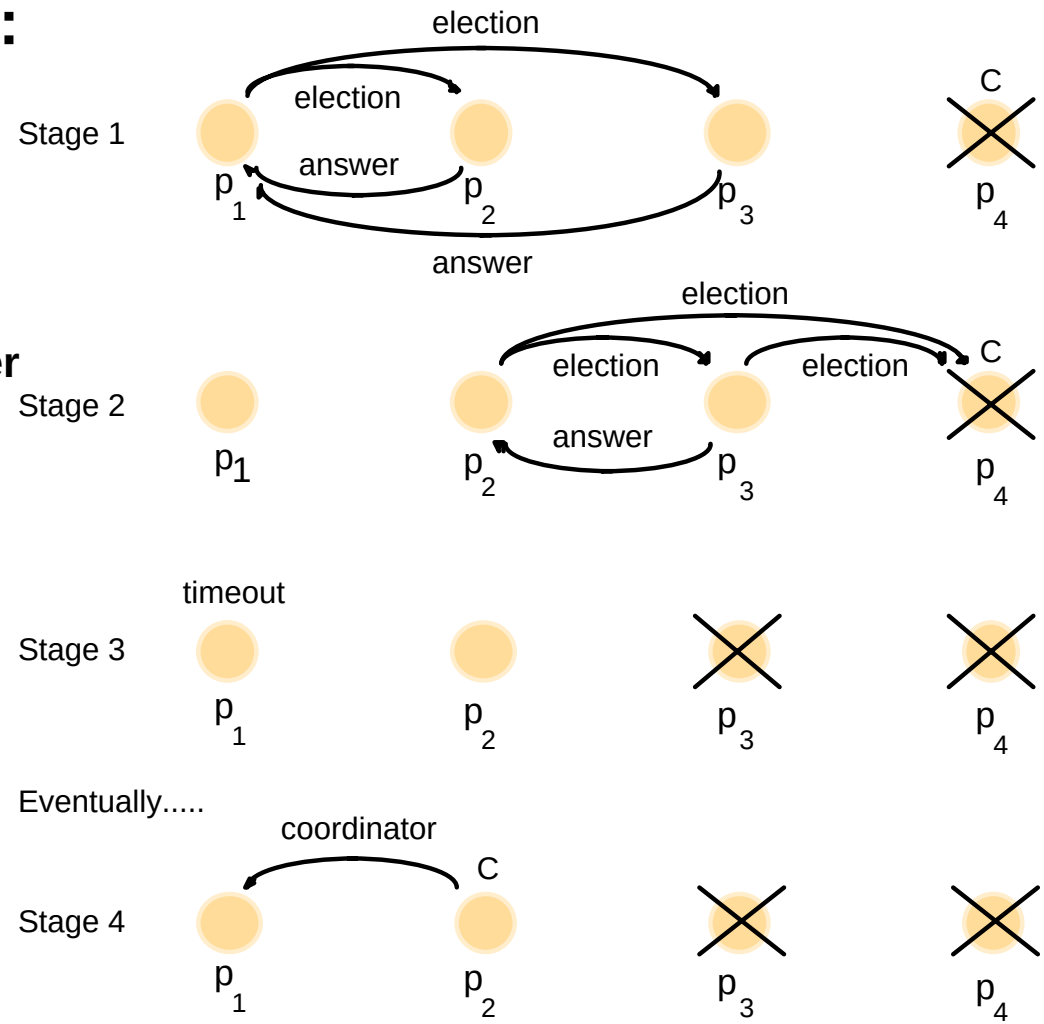
Verfahren für Server, welche alle höherwertigen kennen

(Bully-Algorithmus, Garcia-Molina 1982):

1) Jeder Server testet, ob ein höherrangiger antwortet:

a) Wenn der höherrangige antwortet, unternimmt der Server nichts weiter

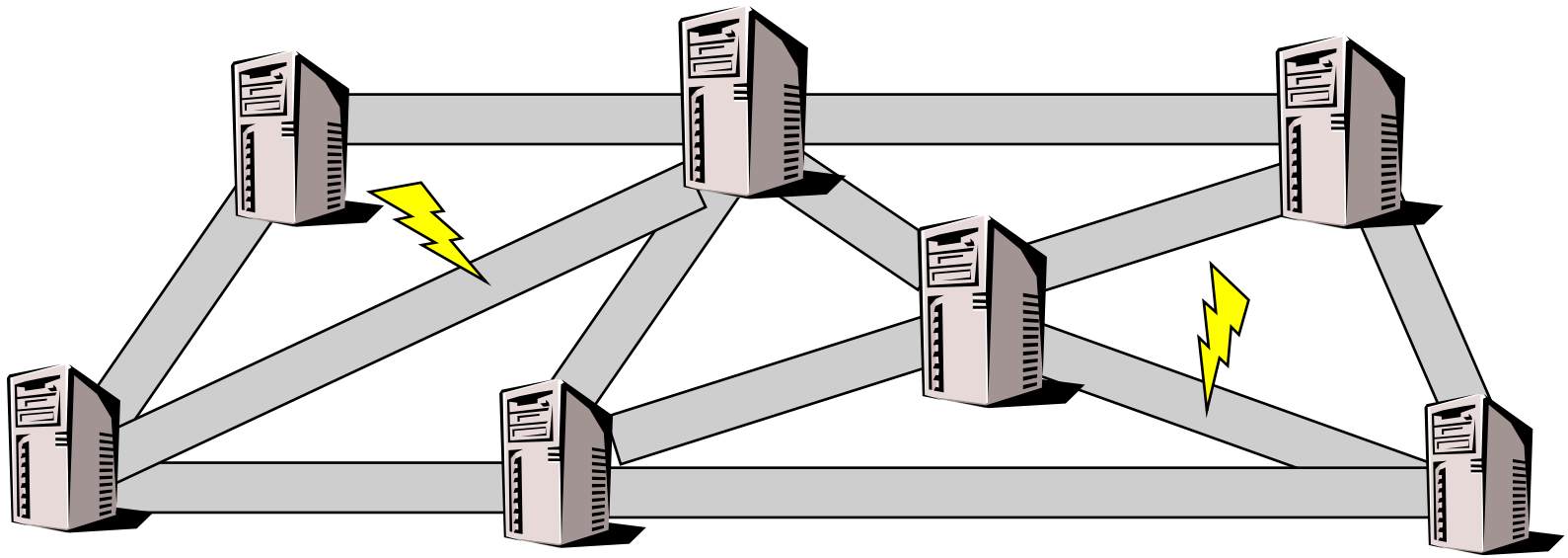
b) Wenn der höherrangige nach timeout nicht geantwortet hat, erklärt sich der Server zum Koordinator und teilt das allen mit, die ihn gefragt haben.



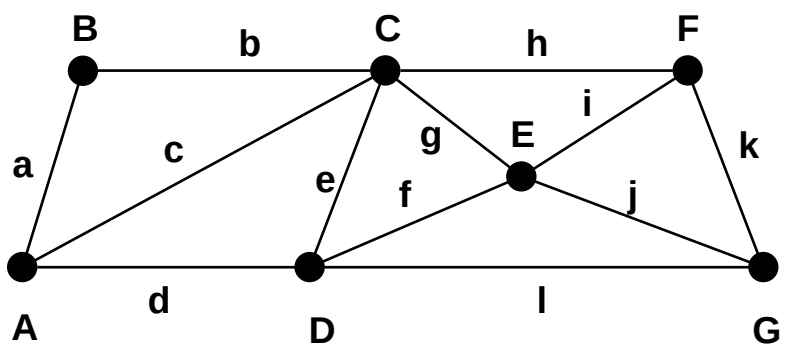
Verteilte Auswahl eines Koordinators

Vergleich Ring-Algorithmus vs. Bully-Algorithmus ?

Behandlung von Leitungsfehlern



Graphentheoretisches Problem:



Gegeben ein Graph mit n Ecken und m Kanten:

- **Finde zu jedem Paar von Ecken (X,Y) einen Weg W aus Kanten des Graphen**
- **Falls eine Kante x ausfällt, ersetze alle Wege W, die x enthalten, durch neue Wege W‘, die x nicht enthalten**

Behandlung von Leitungsfehlern

Verallgemeinertes graphentheoretisches Problem:

Gegeben ein Graph mit n Ecken und m Kanten,
wobei jede Kante x mit einem nichtnegativen Wert bewertet ist:

- **Finde zu jedem Paar von Ecken (X,Y) den kürzesten Weg W aus Kanten des Graphen**

Lösung (Dijkstra):

findet die kürzesten Wege von X zu jeder anderen Ecke in Zeit O(m + n log n)

- **Es ist kein Algorithmus bekannt, der das Problem für feste X und Y schneller löst**

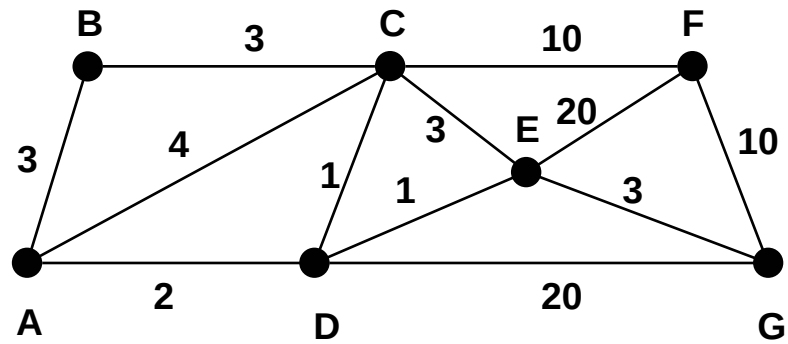
Wenn das Verfahren auf jede Startecke X angewendet wird, dann braucht es insgesamt Zeit $O(nm + n^2 \log n)$.

- **Für die Berechnung aller Wege gibt es etwas schnellere Verfahren**

Vorteil des Dijkstra-Algorithmus:

Eine Variante des Algorithmus lässt sich ohne zentrale Steuerung und ohne globale Kenntnis des Graphen programmieren.

Behandlung von Leitungsfehlern



Leitungsfehler-Problem:

Gegeben ein Graph mit n Ecken und m Kanten,
wobei jede Kante x mit einem nichtnegativen Wert bewertet ist.
Es seien ferner die kürzesten Wege zwischen je zwei Ecken berechnet.

- **Wie kann die Information aktualisiert werden, wenn eine Kante ausfällt ?**

Triviale Lösung:

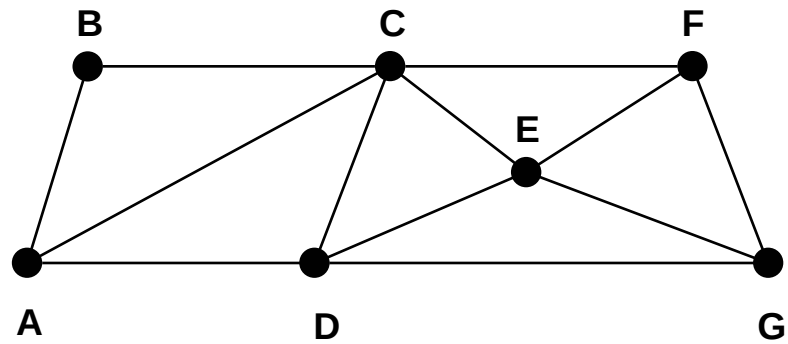
Berechne die kürzesten Wege im veränderten Graphen von neuem !

Frage:

Gibt es eine bessere Lösung ?

Details in Turau, Kapitel 8 (vor allem 8.5)

Bestimmung der Fehlertoleranz



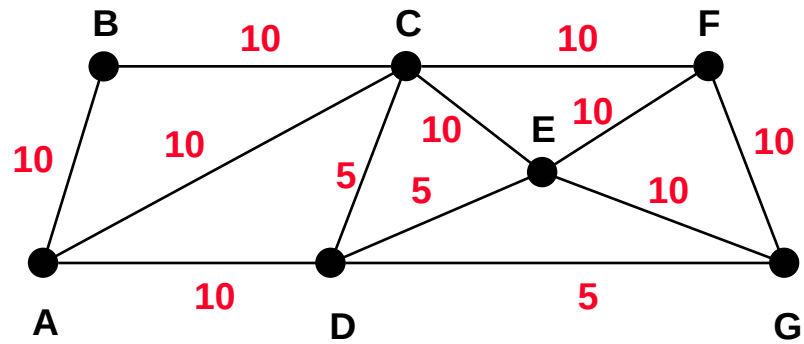
Definitionen:

1) Die Eckenzusammenhangszahl k eines Graphen ist die minimale Zahl von Ecken, die entfernt werden muss, um den Graphen unzusammenhängend zu machen.
2) Die Kantenzusammenhangszahl k eines Graphen ist die minimale Zahl von Kanten, die entfernt werden muss, um den Graphen unzusammenhängend zu machen.
3) Analog werden die Zusammenhangszahlen zwischen zwei bestimmten Ecken eines Graphen definiert

Satz von Menger (1929):

1) Die Eckenzusammenhangszahl k zwischen zwei Ecken X und Y ist gleich der Anzahl verschiedener Wege zwischen X und Y, die paarweise keine Ecke gemeinsam haben.
2) Die Kantenzusammenhangszahl k zwischen zwei Ecken X und Y ist gleich der Anzahl verschiedener Wege zwischen X und Y, die paarweise keine Kante gemeinsam haben.

Bestimmung der Fehlertoleranz



Algorithmen:

1) **Die Eckenzusammenhangszahl k zwischen 2 Ecken X und Y und die zugehörigen eckendisjunkten Wege können in Zeit $O(n^{1/2}m)$ gefunden werden.**

2) **Die Kantenzusammenhangszahl k zwischen 2 Ecken X und Y und die zugehörigen kantendisjunkten Wege können in Zeit $O(n^{2/3}m)$ gefunden werden.**

Hilfsmittel für die Algorithmen:

Algorithmus von Dinic zur Berechnung von maximalen Flüssen in Netzwerken mit Kantenkapazitäten

Details in Turau, Kapitel 6 und 7

Beim nächsten Mal:

WebServices