Verteilte Betriebssysteme

13. Kapitel
Konsens

Matthias Werner
Professur Betriebssysteme

13.1 Einführung

- Wir haben bereits einige Probleme betrachtet, die zum Ziel haben, dass eine Anzahl von Teilnehmern zu einem gemeinsamen Ergebnis kommen, z.B.:
  - Uhrensynchronisation
  - Auswahlproblem
  - Verteiltes Commit
- Ein solches Finden eines gemeinsamen Ergebnisses nennt man **Konsens**

Definition 1 (Konsens)

Konsens erlaubt einer Menge von fehlerfreien Verarbeitungseinheiten den korrekten Wert eines Teils des Systemzustandes zu erfahren und gleichzeitig zu wissen, dass jede andere korrekte Verarbeitungseinheit das selbe Ergebnis erhält, unabhängig von den Aktionen der fehlerhaften Einheiten.

Fehler

- Je nach Annahmen über das System ist Konsens trivial:
  - In einem fehlerfreien System braucht man gemeinsame Information und eine gemeinsame **Bewertungsfunktion**
- **Aber:**
  - sich ändernde Information sind nur zeitweise gültig ➡ Synchronisation erforderlich
  - reale Systeme haben Fehler ➡ Berücksichtigung notwendig
- Hier nehmen wir synchrone Systeme an und betrachten die Auswirkung von Fehlern

Fehler (Forts.)

- Wenn Fehler berücksichtigt werden soll, muss das Verhalten des Fehlers spezifiziert werden
- In einem verteilten System lässt sich das Verhalten **nur** aufgrund von Nachrichten beobachten
- Mögliches Fehlverhalten:
  - Nachrichten kommen nicht oder nicht rechtzeitig an
  - Der Inhalt der Nachricht ist inkorrekt
  - Es werden zusätzliche Nachrichten gesendet
- Ist nur das Kommunikationssystem gestört ist lediglich das erste Verhalten möglich[1], sonst muss der Knoten fehlerhaft sein
- Wenn das Fehlverhalten nicht beschränkt ist, spricht man von einem **Byzantinischen Fehler**
  - Schließt auch **konspiratives** Verhalten ein

[1] typische Fehlertoleranzmaßnahmen wie Checksummen vorausgesetzt

Problem des Byzantinischen Agreements

- Byzantinisches Agreement nach LAMPORT
- Die Armee von Byzanz will mit mehreren Divisionen eine Stadt erobern
- Jede Division wird von einem General befehligt, ein General ist der kommandierende General
- Generäle kommunizieren ausschließlich durch Boten
- Unter den Generälen können sich Verräter befinden
- Eroberung ist nur bei konsistentem Verhalten (der loyalen Generäle) erfolgreich
- **Problem:** Gesucht ist ein Verfahren (Algorithmus), das folgendes garantiert:
  - Alle (loyalen) Generäle befolgen die gleiche Anweisung
  - Ist der kommandierende General loyal, befolgen die (loyalen) Generäle seine Anweisung

Varianten

Es gibt eine Reihe ähnlicher Probleme, die sich aber ineinander überführen lassen:

- **Generalsproblem (Byzantinisches Agreement)**
  - Ein General gibt Befehl; alle loyalen Generäle sollen gleichen Befehl ausführen; wenn der kommandierender General loyal ist, wird sein Befehl ausgeführt.
- **Byzantinischer Konsens**
  - Alle Generäle haben eine private Meinung; alle loyalen Generäle sollen sich auf eine Meinung einigen
- **Interaktive Konsistenz**
  - Alle Generäle haben eine private Meinung; alle loyalen Generäle haben am Ende den gleichen Werte-Vektor, wobei sie Meinungen der loyalen Generäle im Werte-Vektor korrekt sind.

Achtung: Benennung ist in der Literatur unterschiedlich zu finden.

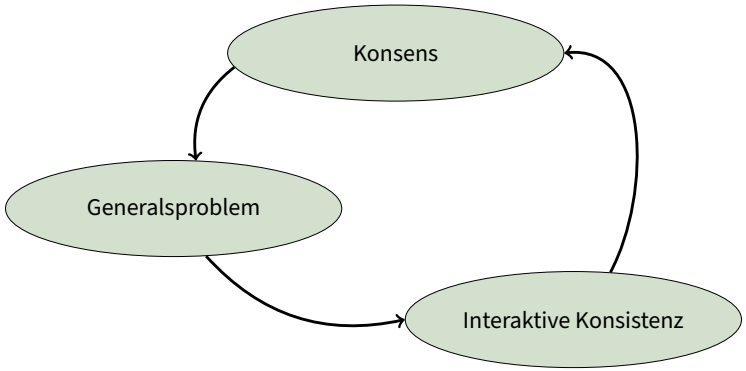
Vergleich

	Generalsproblem	**Konsens**	**Int. Konsistenz**
Startwert(e)	Wert des Kommandants	private Werte	private Werte
Einigung auf	gemeinsamer Wert	gemeinsamer Wert	Wertevektor
Validität (Resultat)	loyaler Kommandant ➡ Wert des Kommandants	$\forall$ korrekte Teilnehmer mit gleichen Wert v ➡ v	korrekter Teilnehmer ➡ korrektes Vektorelement

- Terminierungbedingung ist immer gleich

Äquivalenz

- Jedes Problem kann auf ein anderes zurückgeführt werden



- Daher gelten die meisten Aussage für **alle** Probleme

13.2 Unmöglichkeit von Konsens

Anzahl der „Verräter“

Theorem 2

Wenn f die maximale Anzahl (byzantinisch) fehlerhafter Knoten in einem System S von n Knoten ist, gibt es keinen Algorithmus, der das Konsensproblem löst, solange $n \leq 3f$ gilt

Formale Definition (Konsens)

Eine etwas formalere Definition:

- Eine Menge von n Knoten v_i (Prozessoren, Rechnern, ...) sind durch ein (fehlerfreies) Netzwerk miteinander verbunden. Jeder Knoten besitzt einen zunächst privaten Wert δ_i aus einer Menge Δ möglicher Werte.
- Ziel ist es, dass jeder Knoten für einen Wert aus Δ entscheidet. Drei Bedingungen sollen erfüllt werden:
  - **Agreementbedingung.** Keine zwei (fehlerfreien) Knoten entscheiden sich für unterschiedliche Werte
  - **Validierungsbedingung.** Wenn alle fehlerfreien Knoten mit dem gleichen privaten Wert $\delta \in \Delta$ beginnen, dann muss dies auch der Ergebniswert sein
  - **Terminierungsbedingung.** Alle fehlerfreien Knoten entscheiden sich nach endlicher Zeit

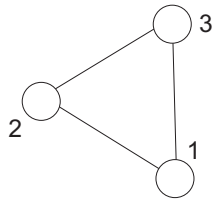
Beweis für Theorem 2, Knotenzahl

- Für $n = 2$: offensichtlich.
- Für $n > 2$: Beweisen zunächst einfacheres Lemma 3

Lemma 3

Das Konsensproblem ist nicht lösbar, wenn $n = 3$ gilt und mindestens ein Fehler möglich ist.

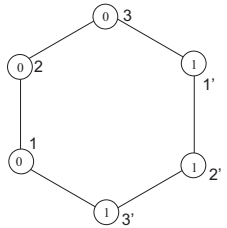
Betrachten System $\mathcal{S}$ mit 3 Knoten:



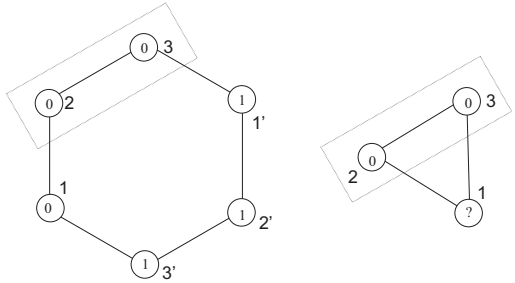
Annahme: In $\mathcal{S}$ lasse sich das Konsensproblem lösen.

Beweis für Lemma 3

- Konstruieren System $\mathcal{S}'$ aus zwei $\mathcal{S}$:

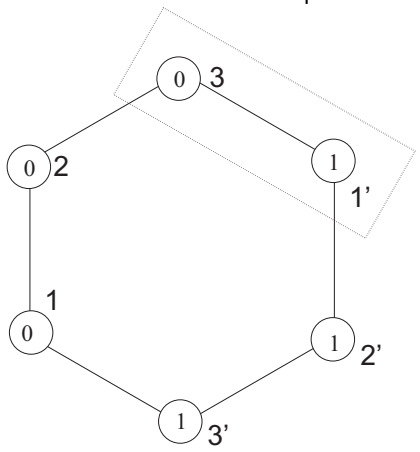


- Für zwei Knoten ist nicht der Unterschied zu $\mathcal{S}$ mit einem defektem Knoten erkennbar



Beweis für Lemma 3 (Forts.)

Betrachtung jeweils zweier anderer Knoten führt zum Widerspruch



Beweis für Lemma 3 (Forts.)

- Für $n \leq 3f$: Annahme des Gegenteils, sei $\mathcal{A}$ ein System mit n Knoten, von denen f fehlerhaft sind.
- A löse das Konsensproblem.
- Partitioniere $\mathcal{A}$ in drei nichtleere Teilmengen I_1, I_2 und I_3 mit $|I_i| \leq f$.
- OBdA: Alle fehlerhaften Knoten von $\mathcal{A}$ sind entweder in I_1, I_2 oder I_3.

Beweis für Lemma 3 (Forts.)

- Betrachten System $\mathcal{B}$, das durch $\mathcal{A}$ simuliert wird.
- $\mathcal{B}$ bestehe aus N_1, N_2 und N_3 (entsprechend I_1, I_2 und I_3)
- $\mathcal{A}$ kann $\mathcal{B}$ durch folgenden Spielregeln simulieren:
  - Alle Knoten in jedem I_i haben den gleichen Startwert v_i
  - Wenn alle Knoten in I_i sich für einen Endwert entscheidet, dann hat sich I_i (d.h. N_i) entschieden. Falls unterschiedliche Werte in I_i entschieden werden, wird einer davon genommen.
- Wenn $\mathcal{A}$ das Konsensproblem löst, wird das Konsensproblem auch für $\mathcal{B}$ gelöst ➡ Widerspruch zu Lemma 3

□

Kommunikation zwischen den Generälen

Theorem 4

In einem System S mit n Knoten, von denen maximal f fehlerhaft sind, gibt es keinen Algorithmus, der das Problem des Byzantinischen Agreements/Konsensproblem löst, wenn die Konnektivität des Kommunikationsgraphen $k = \gamma(S) \leq 2f$ ist.

- Der Beweis von Theorem 4 wird hier nicht geführt ➡ Übung

Bedingungen für Lösbarkeit Byzantinischer Probleme

- Es gibt ziemlich viele Unmöglichkeitsbeweise für das Byzantinische Agreement u.ä. Probleme
- DOLEV ET AL. haben fünf Systemcharakteristika genannt, die eine Rolle spielen:
  - **Ausführung auf den Knoten:** synchron vs. asynchron
  - **Kommunikation:** synchron vs. asynchron
  - **Nachrichtenreihenfolge:** geordnet vs. ungeordnet
  - **Übertragungsmechanismus:** broadcast vs. Punkt-zu-Punkt
  - **Senden/Empfangen-Atomarität:** atomar vs. nicht atomar

Bedingungen für Lösbarkeit Byzantinischer Probleme

Es konnten genau vier Fälle identifiziert werden, bei denen es Byzantinisches Agreement mit $f > 1$ fehlerhaften Knoten geben kann:

- Synchrone Knoten und synchrone Kommunikation
- Synchrone Knoten und geordnete Nachrichtenreihenfolge
- Broadcast-Übertragung und geordnete Nachrichtenreihenfolge
- Synchrone Kommunikation, Broadcast-Übertragung, atomares Senden/Empfangen

Für $f = 1$ gibt es noch einen weiteren Fall:

- Asynchrone Knoten, synchrone Kommunikation, Punkt-zu-Punkt-Kommunikation und atomares Senden/Empfangen

13.3 Algorithmus

- Algorithmus für Generalsproblem (Commander zu Leutnants) mehrfach angewendet
- Annahmen:
  - Sei $|\Delta| = 2$ (z.B. $\Delta = \{a, b\}$)
  - Sei $n > 3 \cdot f$
  - Vollständig vernetzter Graph

Oral-Messages-Algorithmus

1. Kommandierender General sendet an alle Leutnants seinen Befehl
2. Alle senden ihre Sicht an alle anderen
3. Punkt 2 wird f-mal wiederholt
4. Jeder Leutnant führt einen Baum-Mehrheits-Algorithmus durch

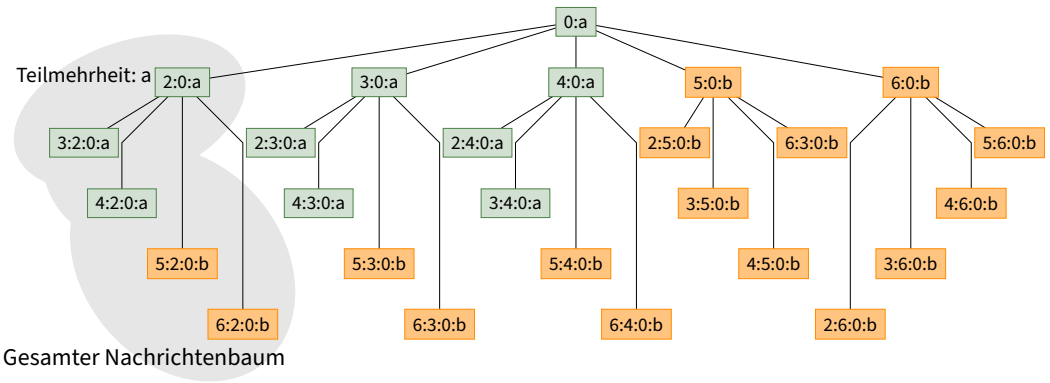
Baum-Mehrheit

Algorithmus

- Jeder Leutnant bildet einen Nachrichtenbaum
- Jede Hierachiestufe ist eine Indirektion der Sicht
- Alle Unterknoten eines Knotens zeigen die (behauptete) Sicht der anderen auf diesen Knoten
- Es wird rekursive die Mehrheit über die Subbäume (Knoten + Unterknoten) gebildet
- Fehlende Nachrichten werden beliebig ersetzt

Beispiel für Baum-Mehrheit, $n = 7, f = 2$

Sicht von Leutnant Nr. 1 (0 ist Kommandierender General):
Nachrichtenformat: Prefix gibt an, wessen Sicht es ist.

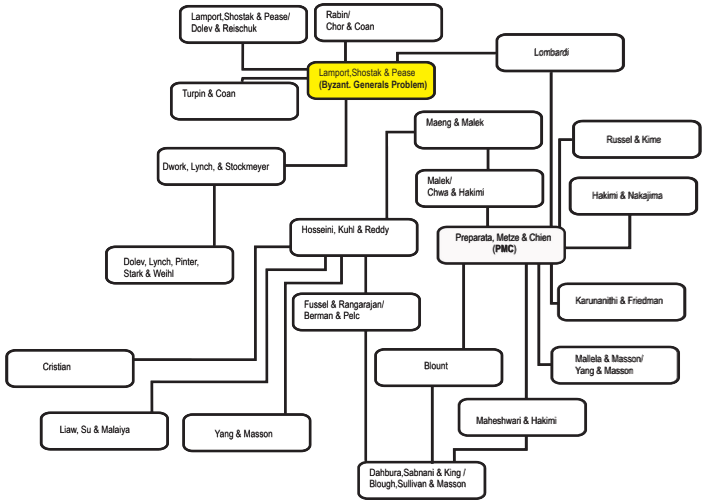


Gesamter Nachrichtenbaum

Effizienz

- **Minimalität**
  - Bewiesen: ein Byzantinisches Agreement braucht mindestens $3f + 1$ Prozessoren, $f + 1$ Runden und Nachrichten in der Gesamtgröße von $\mathcal{O}(f^2)$ Bits
  - Bisher noch kein Algorithmus gefunden, bei dem alle Parameter minimal sind
- **Randomisierte Algorithmen**
  - Erlauben Synchronisationsforderungen aufzugeben
  - Keine Garantie, aber Wahrscheinlichkeit eines Agreements geht gegen 1
  - Im Mittel weniger Runden als deterministische Algorithmen

Verwandschaft zwischen Konsensproblemen



Diskussion

- Obwohl die Annahme Byzantinischer Fehler etwas paranoid wirkt, sind sie in der Realität relevant
  - Uhrensynchronisation ist bei Existenz inkorrekter Uhren **immer** byzantinisch
  - Mehrere Plattformen nutzen Byzantinisches Agreement:
    - SIFT (Software Implemented Fault Tolerance, NASA)
    - TTA
    - BitCoin
    - ...
- Schwächere Annahmen erlauben effizientere Protokolle
  - Authenticated Byzantine fault (oder noch schwächere Fehlermodelle)
  - Keine konsistentes Ergebnis (nur „nahezu“)
  - Keine garantiere Terminierung (probabilistisch)
  - ...

Literatur

[Pra96] Pradhan , D. K. (Hrsg.): *Fault Tolerant Computer Systems*. Prentice Hall, 1996 , Section 3.4 und Chapter 8

[Lyn96] Lynch , N. A. : *Distributed Algorithms*. Morgan Kaufmann, 1996 , Chapter 6

[LSP82] Lamport , L. , R. Shostak und M. Pease: „The Byzantine Generals Problem“. *ACM Transactions on Programming Languages and Systems*, 4(3)1982, 382–401

[BDM93] Barborak , M. , A. Dahbura und M. Malek: „The consensus problem in fault-tolerant computing“. *ACM Computing Survey*, 25(2)1993, 171–220