Verteilte Systeme

Fehlermodelle

Fehlermodelle

Fehler in verteilten Systemen lassen sich grob in drei Kategorien einteilen:

Auslassungsfehler (Omission Errors)

Der Kommunikationskanal lässt Nachrichten aus

Haltefehler (Stopping errors)

Ein Prozess im VS sendet und empfängt ab einem beliebigen Zeitpunkt keine Nachrichten mehr.

Byzantinische Fehler

Ein Prozess lässt Nachrichten aus oder sendet beliebige Nachrichten zu beliebigen Zeitpunkten, ggf. entgegen des Protokolls.

Verteilte Systeme

Verteilte Übereinkunft

Verteile Übereinkunft

Klassisches Setting:

Generäle müssen sich koordinieren. Wenn alle / die Mehrzahl gleichzeitig losschlagen, dann gewinnen sie die Schlacht, wenn weniger gleichzeitig angreifen, werden ihre Armeen vernichtet.

Variante 1: Die Generäle sind alle integer, aber können nur über Boten kommunizieren. Die Zeit, die die Boten zwischen den Lagern brauchen ist bekannt. Es kann passieren, dass die Feinde boten abfangen, und daher Nachrichten verloren gehen -> Auslassungsfehler

Verteile Übereinkunft bei Auslassungsfehlern

Formal: n Prozesse haben einen Startwert c ∈ {0,1}
irgendwann geben alle Prozesse ihre Entscheidung aus.

Forderungen:

Übereinkunft: Alle Prozesse treffen die selbe Entscheidung.

Gültigkeit:
- Wenn alle Prozesse mit c=0 starten, ist die Entscheidung 0
- Wenn alle Prozesse mit c=1 starten, ist die Entscheidung 1

Terminierung: Alle Prozesse treffen irgendwann eine Entscheidung.

Verteile Übereinkunft bei Auslassungsfehlern

Erkenntnis:

Es existiert kein deterministischer Algorithmus, der das “Verteile Übereinkunft bei Auslassungsfehlern”-Problem löst.

Beweisidee:

Angenommen, ein Algorithmus A existiert und entscheidet nach k Runden. Für gegebene Eingaben ist die Zustandsfolge damit fest. Falls in Runde k alle Nachrichten an einen Prozess verlorengehen, dann können die anderen Prozesse dies nicht von einer korrekten Ausführung unterscheiden, und sie geben unterschiedliche Werte aus.

Verteile Übereinkunft bei Auslassungsfehlern

Idee:

Wir bauen uns einen randomisierten Algorithmus, der mit Wahrscheinlichkeit 1-(1/k) korrekt entscheidet.

- Ein Koordinator wählt r ∈ [0,k] und verteilt r.
- Wir verwenden einen Skalar “level” um den Zustand der anderen zu Verfolgen.
- Wenn nach k Runden level > r, dann wird das Ergebnis des deterministischen Algorithmus verwendet, ansonsten 0 ausgegeben.
- Falls nach k runden level = r ist, können Prozesse im level um 1 daneben liegen

=> Wahrscheinlichkeit für Fehlentscheidung: 1/k

Verteile Übereinkunft

Klassisches Setting:

Generäle müssen sich koordinieren. Wenn alle / die Mehrzahl gleichzeitig losschlagen, dann gewinnen sie die Schlacht, wenn weniger gleichzeitig angreifen, werden ihre Armeen vernichtet.

Variante 2: Die Generäle sind alle integer und können zuverlässig über Boten kommunizieren. Die Zeit, die die Boten zwischen den Lagern brauchen ist bekannt. Es kann passieren, dass ein Lager vom Feind überrannt wird, und irgendwann nicht mehr teil nimmt.

Verteile Übereinkunft bei Haltefehlern

Formal: n Prozesse haben einen Startwert c
Alle Prozesse sind initial lebendig, f Prozesse stellen, wenn sie ein stop-Signal bekommen, ihre Tätigkeit ein.
irgendwann geben alle lebendigen Prozesse ihre Entscheidung aus.

Forderungen:

- Übereinkunft: Alle lebendigen Prozesse treffen die selbe Entscheidung.
- Gültigkeit: Falls alle Prozesse den gleichen Startwert c haben, ist die Entscheidung c.
- Terminierung: Alle Prozesse entscheiden irgendwann.

Verteile Übereinkunft bei Haltefehlern

Voraussetzung:

Jeder Prozess p hat einen Startwert $c_p \in C$.

Jeder Prozess hat eine Entscheidung W, inital $W=c_p$

Idee:

Jede Runde flutet p sein W und für jede eingehende Nachricht m setzt p sein $W = W \cup m$.

Korrektkeit: nach f+1 Runden haben alle Prozesse dasselbe W

Verteile Übereinkunft bei Haltefehlern

Zeitkomplexität:

$O(f+1)$

Kommunikationskomplexität:

- f+1 Runden
- b = log(|C|) bits für die Elemente von C
- n*b bits pro Nachricht
- n-1 Nachrichten pro Knoten pro Runde => $O(n^2)$
  => $O(n^3 b)$

Verteile Übereinkunft

Klassisches Setting:

Generäle müssen sich koordinieren. Wenn alle / die Mehrzahl gleichzeitig losschlagen, dann gewinnen sie die Schlacht, wenn weniger gleichzeitig angreifen, werden ihre Armeen vernichtet.

Variante 3: Generäle können zuverlässig über Boten kommunizieren. Die Zeit, die die Boten zwischen den Lagern brauchen ist bekannt. Es gibt f Verräter unter den Generälen, die das Protokoll nicht einhalten, d.h. sie senden beliebige oder keine Daten.

Verteile Übereinkunft bei byzantinischen Fehlern

Formal: n Prozesse haben einen Startwert c
irgendwann geben alle nicht byzantinischen Prozesse ihre Entscheidung aus.

Forderungen:

- Übereinkunft: Alle ehlichen Prozesse treffen die selbe Entscheidung.
- Gültigkeit: Falls alle Prozesse den gleichen Startwert c haben, ist die Entscheidung c.
- Terminierung: Alle Prozesse entscheiden irgendwann.

Verteile Übereinkunft bei byzantinischen Fehlern

Erkenntnis:

Es existiert kein Algorithmus, der das “Verteile Übereinkunft bei byzantinischen Fehlern”-Problem bei n<3f löst.

Beweis:

s. Lynch, S. 116ff

Verteile Übereinkunft bei byzantinischen Fehlern

Erkenntnis:

Es existiert kein Algorithmus, der das “Verteile Übereinkunft bei byzantinischen Fehlern”-Problem bei n<3f löst.

Beweis:

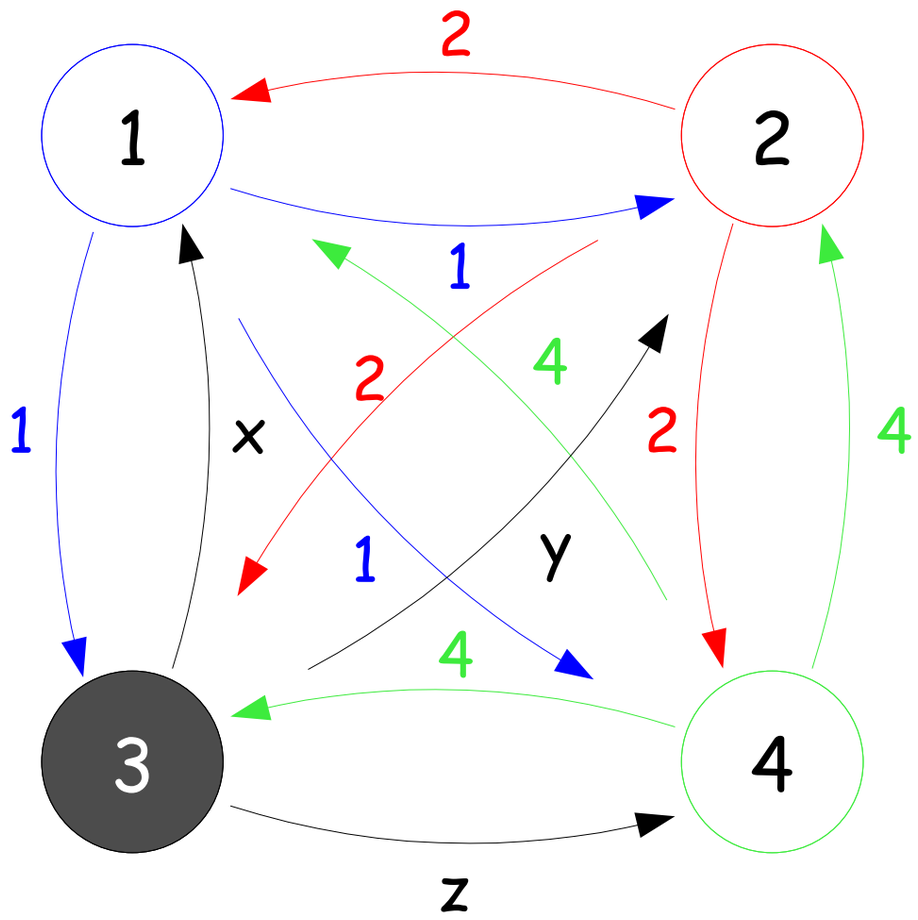
s. Lynch, S. 116ff

Verteile Übereinkunft bei byzantinischen Fehlern

Idee:

Wir bauen einen Algorithmus, der die Initialwerte der anderen Knoten und die annahmen über die Initialwerte zuverlässig verbreitet. Wir können dann daran die fehlerhaften Prozesse an Widersprüchen erkenne.

Verteile Übereinkunft bei byzantinischen Fehlern



4 byzantinische Generäle

1. Schritt:
Nachrichtenaustausch

2. Schritt (lokal):
Akkumulieren
1 bekommt (1,2,x,4)
2 bekommt (1,2,y,4)
3 bekommt (1,2,3,4)
4 bekommt (1,2,z,4)

3.Schritt: Vektoren austauschen

1 bekommt:	2 bekommt:	4 bekommt:
(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)

4. Schritt (lokal):
Konsens ermitteln
(1,2,UNKNOWN,4)

Verteile Übereinkunft bei byzantinischen Fehlern

Verallgemeinerung für f>1:

Falls wir f>1 tolerieren müssen, verallgemeinern wir das Konzept:

- Wir tauschen senden f Runden lang an alle Knoten, welche Vektoren wir in der letzten Runde erhalten haben.
- Die Werte werden in einem Baum gespeichert, so dass es zu jedem Weg, den ein Initialwert genommen hat, einen Weg im Baum gibt.
- Wir laufen den Baum von den Blättern aufwärts und übernehmen auf jeder Ebene k den Wert der Mehrheit der Knoten auf Ebene k+1

Verteile Übereinkunft bei byzantinischen Fehlern

Zeitkomplexität:

$O(f+1)$

Kommunikationskomplexität:

$O(n^{f+1}b)$

=> Sehr teuer!

Es existieren Algorithmen mit n2-Laufzeit, falls man die Nachrichten signiert (d.h. Prüfen kann ob sie korrekt weitergeleitet wurden).

Verteile Übereinkunft im asynchronen Fall

Es existiert ein Unmöglichkeitsbeweis sowohl für Haltefehler, als auch für byzantinische Fehler!

Einzige Möglichkeit: Runden durch Timeouts generieren!

k-Agreement

Häufig genügt es, dass nur k<n Stationen sich auf einen Wert einigen (k-Agreement).

Analog zur Übereinkunft lösbar

Quorum

Statt eine Übereinkunft zu realisieren, wird eine Zustimmung von w Stationen angefordert (z.B. zum Schreiben eines Datums).

Idee: n-f < w < 2n => Ausschluß von Widersprüchen

Verallgemeinerung:

Leser/Schreiber Ausschluß:

zum Lesen werden nur n-f < r Zustimmungen gefordert, wobei für Leser/Schreieber-Ausschluss r+w > n gilt.