Byzantinische Fehler

Matthias Reumann

8. April 2020

Inhaltsverzeichnis

1 Byzantinische Fehler

1.1 Definition

Byzantinische Fehler treten in verteilten Systemen auf, wenn fehlerhafte Nachrichten von einem Knoten versendet werden, die von der Struktur und vom Inhalt nicht von fehlerfreien unterscheidet werden können. [2]

Den Namen hat diese Fehlerklasse im Jahr 1982 von Lamport et. al. im Papier *The Byzantine Generals Problem* erhalten.[1] In diesem werden Byzantinische Fehler durch das abstrakte Beispiel der Byzantinischen Generäle näher erklärt.

1.2 Anwendungen

Da eine Fehlentscheidung katastrophale Folgen hätte, wird in Boeing Flugzeugsystemen Byzantinische Fehlern eine hohe Bedeutung zugewiesen. Das Raumfahrtunternehmen SpaceX setzt auch auf BFT in ihren Dragon Raumschiffen, welche Astronauten zur ISS transportieren sollen. Die Cryptowährung Bitcoin verwendet auch BFT Implementierungen um die Integrität ihrer Blockchain zu gewährleisten.

2 Problem der byzantinischen Generäle

Mehrere Truppen mit jeweils einem General umzingeln eine feindliche Stadt und kommunizieren direkt über Boten miteinander. Jeder der Generäle beobachtet den Feind und schlussendlich müssen die Generäle eine gemeinsame Entscheidung treffen. Jedoch kann es unter den Generälen Verräter geben, deren Ziel es ist eine gemeinsame Entscheidung der loyalen Generäle zu unterbinden.

Die Generäle müssen einen Algorithmus finden der garantiert, dass

(a) Alle loyalen Generäle die gleiche Entscheidung treffen

(b) Eine geringe Anzahl an Verrätern die loyalen Generäle nicht zu einer Fehlentscheidung führt

Eine Entscheidung unter den Generälen wird gefällt indem jeder General den Feind beobachtet und seine Informationen an die anderen Generäle weitergibt. *v(i)* ist die Nachricht gesendet vom i.ten General. Jeder General erhält die Nachrichten *v(1)* bis *v(n)*, wobei n die Anzahl der Generäle darstellt. Die finale Entscheidung wird durch die absolute Mehrheit der Werte dieser Nachrichten bestimmt. Allerdings könnte es sein, dass loyale Generäle unterschiedliche *v(i)* erhalten, da ein Verräter unterschiedliche Werte zu unterschiedlichen Generälen sendet, was widerum Punkt a widerspricht.

Deshalb müssen sogennante *interactive consistency conditions* gelten. Ein führender General, auch Commander genannt, sendet einen Befehl an seine $n-1$ Leutnant Generäle so, dass:

IC1 Alle loyalen Leutnants den gleichen Befehl ausführen
IC2 Wenn der Commander loyal ist, dann befolgt jeder loyale Leutnant seinen Befehl

2.1 Beispiel mit drei Generälen

In diesem Abschnitt wird eine Situation beschrieben in der drei Generäle miteinander kommunzieren. Wichtig dabei ist, dass niemand der loyalen Generäle weiß wer der Verräter ist.

In Abbildung 1 ist Leutnant 2 (L2) der Verräter. Der loyale Commander sendet den Befehl *ANGRIFF* an beide Leutnants L1 und L2. Zur Verifizierung sendet L2 L1 die Nachricht, er habe den Befehl *RÜCKZUG* erhalten. L1 hat nun die Nachrichten *ANGRIFF* und *RÜCKZUG* und kann auf Grund dessen keine eindeutige Entscheidung treffen.

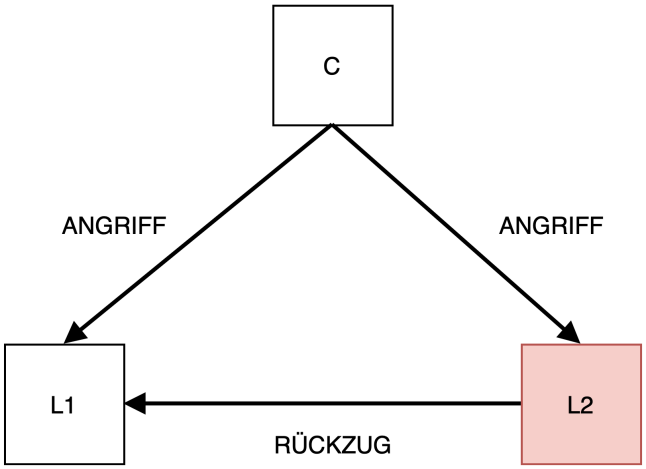

Abbildung 1: L2 ist der Verräter. L1 muss den Befehl des Commanders auführen um die IC2 einzuhalten.

Im nächsten Szenario (Abbildung 2) ist der Commander der Verräter. Dieser sendet unterschiedliche Befehle an L1 und L2. Für L1 ist es die exakt selbe Situation wie in Abbildung 1. L2 steht vor dem selben Problem.

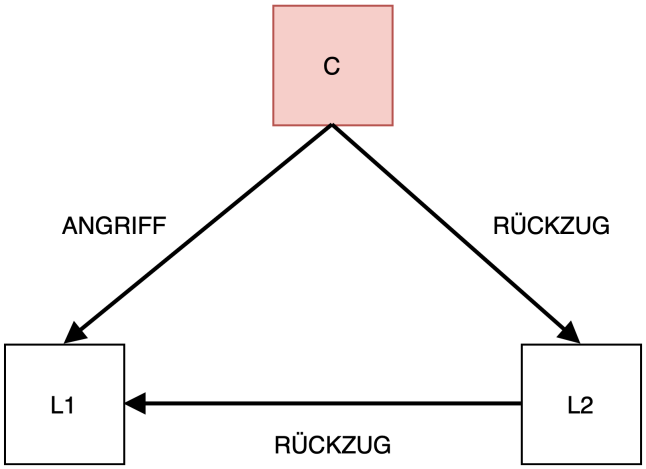

Abbildung 2: Der Commander ist der Verräter. Für L1 ist es die selbe Situtation wie in Abbildung 1.

Wie aus diesem Beispiel hervorgeht, existiert keine Lösung für drei Generäle mit einem Verräter. Um einen byzantinischen Fehler zu erkennen muss daher folgende Eigenschaft gelten: $n \geq 3m + 1$, wobei n die Anzahl der Generäle und m die Anzahl der Verräter in diesem System ist.

2.2 Lösung

Eine mögliche Lösung dieses Problems bietet der Oral-Message-Algorithmus, kurz OM-Algorithmus. Der Algorithmus funkioniert für Systeme die die Eigenschaft $n \geq 3m + 1$ erfüllen. Außerdem werden folgende Annahmen getroffen:

- Jede Nachricht wird korrekt empfangen
- Der Empfänger kann sicherstellen von wem die Nachricht versendet wurde
- Fehlende Nachrichten können erkannt werden

Der Algorithmus funktioniert nur wenn ein General mit jedem anderen General direkt miteinander kommunzieren kann. Ein Verräter-Commander könnte auch keinen Befehl an seine Leutnants senden, daher ist der Standardbefehl in diesem Fall *RÜCKZUG*.

Listing 1: Algorithmus OM(0)

```
1) Commander sendet seinen Wert v0 an alle Leutnants
2) Jeder Leutnant verwendet den Wert v0 oder
   falls dieser nicht vorhanden ist den Standardbefehl
```

Listing 2: Algorithmus OM(m) m > 0

```
1) Commander sendet seinen Wert v0 an alle Leutnants
2) Für jedes i gilt:
     vi := Wert den General i vom Commander erhalten hat
     (oder Standardbefehl)
     General i agiert wie der Commander im OM(m-1) um seinen
     Wert vi den anderen n-2 Generälen zu senden

3) Für jedes i mit j != i gilt:
     vj := Wert den General i von General j in Schritt 2 erhalten hat
     General i benutzt die Funktion majority(v1, vn-1)
```

Die Funktion *majority* ermittelt die absolute Mehrheit. Es muss nicht immer die absolute Mehrheit als Entscheidungsfunktion verwendet werden.

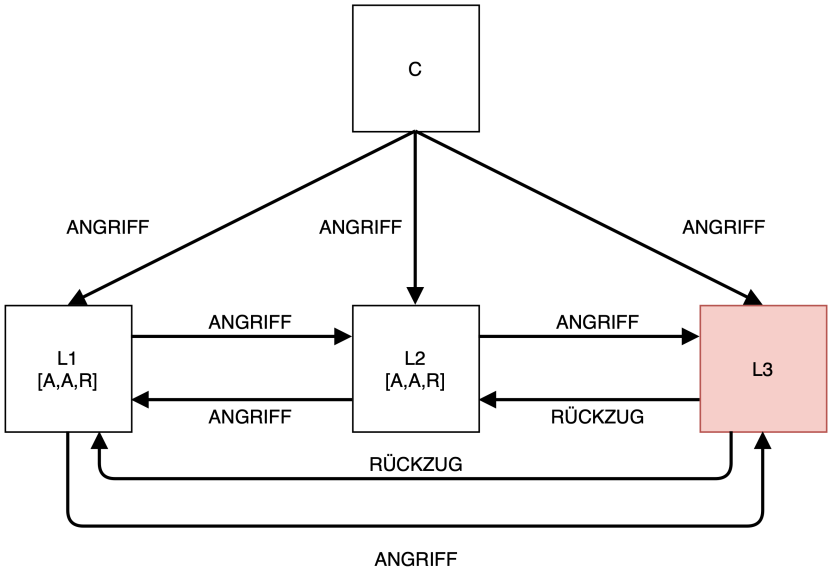

Abbildung 3: L3 ist der Verräter. Der Algorithmus garantiert, dass beide loyalen Leutnants (L1 und L2) die gleiche Entscheidung treffen: *ANGRIFF*.

2.3 Kritik

Der OM-Algorithmus ist sehr aufwendig. Er ist praktisch nicht für große n und m einsetzbar.