Freie Universität Berlin
Institut für Informatik

# Seminararbeit
# Byzantinische Fehler

erstellt von:
Esra Ünal
Matrikelnummer:(4308463)

Proseminar Technische Informatik

# Inhaltsverzeichnis

# 1 Einleitung

Motorsteuerungen, Flugplanungen, Kernreaktorsteuerungssysteme, und Herz-Lungen-Maschinen sind einige von vielen Beispielen für zuverlässige Systeme, deren Bedeutung im Leben vieler Menschen stetig steigt. Charakteristisch für solche Systeme ist ihre Sicherheit und Zuverlässigkeit gegenüber schwerwiegenden Fehlern, denn diese hätten gravierende Konsequenzen, bei denen sogar die Gesundheit von Menschen in Gefahr steht. Viel komplexer wird der Umgang mit diesen Konstruktionen jedoch, wenn es sich dabei um verteilte Systeme handelt, denn die Komplikationen einer Fehlerbehandlung nehmen mit der Anzahl an Bestandteilen zu.
Verteilte Systeme sind nämlich dafür bekannt, dass sie aus einer Menge von Komponenten bestehen, die alle jeweils mit einem bestimmten Teil an der Gesamtaufgabe des Systems beschäftigt sind und zusätzlich für die korrekte Ausführung mit allen anderen Mitgliedern interagieren und koordinieren müssen. Typischerweise werden hierfür oft Prozessor- oder Servernetzwerke als Beispiel gegeben. Neben diesen gibt es aber auch noch viele weitere Systeme wie Sensoren, die zu einem Netzwerk zusammengefasst werden können, um bestimmte Forderungen zu erfüllen, die mit einer Einheit allein nicht realisierbar sind. Zu dem besitzen sie einen sehr großen Einsatzgebiet von dem eine unter Abschnitt eins beschrieben wird.
Bei solchen komplexen Gebilden müssen folglich sowohl die Einzelbestandteile als auch das Gesamtgerüst als potentielle Fehlerquelle in Betracht gezogen werden, wenn die Sicherheit optimiert werden soll.
Um den gewünschten Schutz vor solchen Systemversagen zu garantieren, müssen daher alle relevanten kritischen Fehlerklassen ausgeschlossen werden.

Ein Aspekt, welches in dieser Arbeit eine besondere Rolle spielt, sind Fehler, wo aufgrund der Weitergabe von fehlerhaften Ergebnissen verursacht von einer Komponente im System in Folge einer Fehlfunktion oder Manipulation, die dann von anderen Systembestandteilen genutzt werden, sind Probleme mit denen zuverlässige Systeme umgehen müssen. Der größte Hindernis bei diesen Defiziten ist die Tatsache, dass die Struktur und der Inhalt der verfälschten Daten äußerlich von den richtigen nicht unterscheidbar sind und daher die gesamte Systemausführung unbemerkt beeinträchtigen.
Viel beunruhigender ist allerdings die Beobachtung, dass diese Fehler viel zu oft unterschätzt werden. Obwohl das Problem mit der entsprechenden Lösung bereits im Jahre 1982 erstmals von unter anderem Lamport [LSP82] der Öffentlichkeit geboten wurde, gibt es dennoch zahlreiche neu entwickelte Systeme, die die Byzantinische Fehlerklasse vernachlässigen. Sei es wegen dem zu hohem Aufwand für die Bereitstellung der Lösung oder aufgrund der weitverbreiteten Meinung, dass die Häufigkeit von Byzantinischen Fehlern verschwindend gering sind, bleibt die Fehlerklasse oft unberücksichtigt. In [DHSZ], in dem ebenso dieses Verhalten angeprangert wird, ist jedoch gezeigt, dass die Häufigkeit beachtlich hoch ist und daher eine bedeutende Relevanz hat.
Diese Art der Fehlerquelle, genannt Byzantinische Fehler, werden nun im Folgenden in dieser Arbeit näher betrachtet. Unter Kapitel zwei wird eine Anwendung eines Sensornetzwerkes gezeigt, wo der Ursprung und die Entwicklung des Fehlers anhand des Beispiels näher beschrieben wird. Anschließend wird in Kapitel drei die Fehlerklasse formal definiert und bereits Voraussetzungen für eine geeignete Lösung mittels des vorgestellten Beispiels unter Kapitel zwei herausgearbeitet. Die tatsächliche Lösung mit dem entsprechenden Algorithmus wird im Abschnitt vier erfasst und zusammen mit anderen Optimierungsvorschlägen im Kapitel fünf erweitert.

# 2 Anwendung

Sensornetzwerke sind Konstruktionen, die in vielen Bereichen des Lebens Anwendung finden. Angefangen von wissenschaftlichen Beobachtungen eines Naturgeschehens bis hin zu Kontrolle des Verkehrsflusses im Straßennetz oder auch gängige Überwachungsanlagen von Gebäuden sind auf Techniken angewiesen, bei denen grundsätzlich eine Vielzahl von Sensoren eingesetzt werden. Betrachtet wird nun im Folgenden eine vereinfachte Feuermeldeanlage für ein Bürogebäude, der bei einem Brand einen Alarm auslöst und zusätzlich Wasser in die betroffene Region sprüht.

Hierfür wird ein großes Sensornetzwerk mit Funkübertragung angelegt. Das Netzwerk bestehend aus $n$ Sensoren dient zur Aufnahme von hohen Temperaturen, die durch einen ausgelösten Feuer verursacht werden, zur Übermittlung dieser Informationen an alle anderen Sensoren, um eine gemeinsame Entscheidung zu fällen, ob tatsächlich ein Brand ausgelöst wurde oder es sich nur um eine Fehlbeobachtung handelt.
Registriert ein Sensor $i$ einen hohen Temperaturwechsel, dann wird der gemessene Wert zunächst mit einem Schwellwert $x$ verglichen und erst bei Überschreitung von $x$ dies den anderen $n-1$ Sensoren und einer zentralen Steuerung mitgeteilt, damit eine Vereinbarung getroffen bzw. eine Warnung ausgegeben werden kann. Folglich sind die Knoten in der Lage Daten mit genau zwei verschiedenen Inhalten zu übermitteln.
Eine neue Nachricht $\mathrm{A}_1$ repräsentiert, dass ein nicht regulärer Temperaturunterschied wahrgenommen wurde, wohingegen $\mathrm{A}_0$ den Normalfall ohne Veränderungen charakterisiert.
Die Aufgabe der Sensoren ist es nun ausgehend von den angekommenen binären Daten eine Entscheidung zu fällen, ob ein Alarm ausgelöst oder die Überwachung fortgesetzt wird. Die Wahl basiert erwartungsgemäß auf der absoluten Mehrheit der gesendeten Nachrichten, d.h. die Sprinkleranlage wird gestartet, wenn die meisten Sensoren einen Brand registriert haben und sonst nicht.
Diese Konstruktion ist solange zuverlässig wie die übermittelten Informationen richtig sind, da keines der Sensoren die Möglichkeit besitzt, den Wahrheitsgehalt der Daten zu prüfen. Tendiert man nämlich aufgrund einer Fehlfunktion oder sogar einer absichtlichen Verfälschung einiger Sensoren zu einer falschen Entscheidung wie zum Beispiel keinen Alarm auszulösen, obwohl einer erforderlich ist, können fatale Konsequenzen entstehen.
Angefangen von der finanziellen Last, die aus der Beschädigung der gesamten Büroausstattung resultiert, falls die Sensoren die Ventile lösen und Wasser sprühen, obwohl kein Feuer entfacht ist, bis hin zur Frustration der Bewohner und Nutzer dieser Behausung, ist ein Fehler nicht erwünscht. Wenn die Sensoren jedoch trotz eines Brandes nichts unternehmen, dann kann das vielen Menschen das Leben kosten und bestehende Arbeitsplätze vernichten. Hinsichtlich der Nutzung eines solchen Systems für andere Anwendungen sind die Folgen ähnlich wie beim Brandmelder außerordentlich hoch. In typischen Ad-hoc- Netzwerken zum Beispiel ist die Lage ein wenig zugespitzter, denn alle Sensoren in diesem temporär erstellten Netzwerk sind auf ihre jeweiligen Nachbarn angewiesen, weil ihre Fähigkeiten allein nicht reichen, die gewünschte Informationen den anderen Mitgliedern des Systems zu melden. Wenn dann eines dieser Sensoren fehlerhaft ist, so vervielfacht sich auch die Fehlerrate um den Faktor der Sensoren, die den defekten Teilnehmer für die Übertragung nutzen. Somit werden auch ihre ursprünglich korrekten Daten falsch übertragen. Nimmt man auch die Tatsache zu Kenntnis, dass viele dieser Überwachungen sich auch auf kritische Bereiche erstrecken, wo Sensoren eingesetzt werden, um Autobahnen, Flughäfen, Kraftwerke und Produktionsanlagen effektiv zu kontrollieren, so wird die Bedeutung dieser Fehler sehr klar.
Folglich können nicht funktionstüchtige Sensoren eines Netzwerkes, die nicht unterscheidbare falsche Inhalte liefern,das System zwingen, fehlerbehaftete Informationen als Grundlage für eine wichtige Entscheidung zu nehmen und daher eventuell die entgegengesetzte Wahl zu treffen oder sogar das gesamte System zum Erliegen zu bringen.
Systemversagen deren Ursache auf die kontinuierliche Verwendung von falschen, aber dennoch protokollkonformen Ergebnissen, die durch Systemkomponenten hervorgehoben werden, zurückzuführen sind, bezeichnet man als byzantinische Fehler. Die folgenden Kapiteln geben einen Einblick in diese Fehlerklasse.

# 3 Problemdefinition

Den Namen Byzantinische Fehler erhielt die bereits unter Kapitel 2 eingeführte Fehlerklasse von dem Begründer dieser Fehler. Zum ersten mal wurde die Fehlerklasse nämlich im Jahre 1982 von Lamport, Shostak und Pease im Papier *The Byzantine Generals Problem* [LSP82] aufgegriffen und gründlich analysiert.
In der Publikation wurde für die verständliche Darlegung des Problems der Fehler stark abstrahiert und auf ein fiktives Beispiel aus einem anderen nicht technischen Bereich übertragen, und zwar auf byzantinische Generäle. Basierend auf dieser Abstraktion wurde in der Ausarbeitung

die Fehlerklasse als Byzantinische Fehler bezeichnet, was sich bis heute in der Literatur bewährt hat. Das Modell beschreibt, dass den Heeresführern aus der Byzanz der Befehl erteilt wurde, eine feindliche Stadt anzugreifen. Aus koordinatorischen Gründen umzingeln diese die Stadt und sind aufgrund dessen räumlich voneinander getrennt. Anschließend warten sie in ihren Postionen auf den geplanten Angriff, der mithilfe von Boten übermittelten Nachrichten einstimmig entschieden werden soll.
Zu der räumlichen Distanzierung kommt noch die Schwierigkeit hinzu, dass unter den Befehlshabern Verräter vorhanden sind, die die Ausführung des Plans verhindern wollen und daher Informationen mit absichtlich falschen Inhalten versenden. Das beschriebene Modell beschreibt genau die Art von Fehlern, die zuvor im Beispiel mit der Brandschutzanlage bereits vorgestellt wurde. Die Sensoren werden von den Generäle repräsentiert und umfassen durch die sogenannten Verräter auch all diejenigen Sensoren, die fehlerhaft funktionieren. Die in dieser Darstellung getroffene Entscheidung über einen künftigen Angriff entspricht übertragen auf die Sprinkleranlage der Wahl zwischen dem Auslösen eines Alarms oder der Fortsetzung der Temperaturüberwachung.
Zusammengefasst bezeichnen Byzantinische Fehler beliebige Fehler, die bei der Ausführung eines in verteilten Systemen eingesetztes Algorithmus auftauchen kann [910]. Für die verständliche Illustration der nachfolgenden Sachverhalte wird von nun an die Logik anhand des Beispiels mit den Sensoren beschrieben und auf das Modell mit den byzantinischen Heeresführern verzichtet, weil sie abstrahiert denselben Kernpunkt widerspiegeln.
Reduziert man das Problem, ausgehend vom Sensornetzwerk, auf eine wesentliche formale Struktur, so erhält man zunächst $n$ Knoten, die die Sensoren repräsentieren. Diese kommunizieren mithilfe von Nachrichten $\mathrm{v_i}$, wobei $\mathrm{v_i}$ die vom i-ten Sensor gesendete Information darstellt.
Die absolute Mehrheit der Elemente in der Sequenz $\mathrm{v_0}$,...,$\mathrm{v_n}$ verrät dann erwartungsgemäß die Entscheidung bzw. das Ergebnis wie die Fortführung des Gesamtsystems aussieht. Zusätzlich zu diesen Bezeichnungen werden zwei Voraussetzungen aufgestellt, deren Einhaltung für die geeignete Lösung essentiell sind. In der Arbeit von Lamport werden diese als *interactive consistency conditions* bezeichnet.

**IC1**: Alle funktionstüchtigen Sensoren treffen dieselbe Entscheidung
**IC2**: Wenn der Mastersensor korrekt funktioniert, dann befolgen alle intakten Sensoren seine Entscheidung

Der Mastersensor, ist derjenige Mitglied im System, der die gravierende Temperaturdifferenz als erster registriert und beginnt alle Sensoren zu benachrichtigen und startet folglich das Austauschen der Informationen zwischen den Komponenten. Es ist an dieser Stelle anzumerken, dass IC2 IC1 induziert, falls der erste Sensor korrekt arbeitet.
Im nächsten Schritt wird mit der Suche nach einer oberen Schranke für die Anzahl an fehlerhaften Teilnehmern im System, die den Fortschritt des Gesamtsystem nicht behindern, begonnen. Erwartungsgemäß ist die ermittelte Zahl abhängig von dem Anteil an korrekt funktionierenden Bestandteilen im System.
Ob Fehlertoleranz im Szenario mit genau einer falschen Komponente und einem korrekt funktionierenden Teilnehmer möglich ist, ergibt sich aus dem Sachverhalt heraus und ist daher trivial. Bei dem konstruierten Fall bestehend aus einem fehlerhaften Mitglied und zwei weiteren normalen Bestandteilen ist das Problem schon ein wenig komplexer und nicht auf Anhieb bestimmbar.
Hierfür ist auch zunächst eine Unterscheidung zwischen den Sensoren zu treffen, die Mängel aufweisen. Wenn der Mastersensor nämlich fehlerhafte Informationen produziert und es den anderen Sensoren übermittelt, so hat weder Sensor $\mathrm{s_1}$ noch Sensor $\mathrm{s_2}$ die Chance zu entscheiden, welches Datum sie für künftige Aufgaben verwenden sollen. Laut Abbildung 1 wird $\mathrm{s_2}$ dem Sensor $\mathrm{s_1}$ berichten, dass es Datum $B$ vom Master erhalten hat, was aber zum Widerspruch zu dem Datum steht, den $\mathrm{s_1}$ vom Mastersensor bekommen hat. Der Sensor $\mathrm{s_2}$ steht analog vor der selben Entscheidung.
Unter der Annahme, dass $\mathrm{s_1}$ oder $\mathrm{s_2}$ das System mit mangelhaften Informationen beliefert, erhält man das gleiche Ergebnis, denn dann würde beispielsweise $\mathrm{s_2}$ mitteilen, dass das Datum $B$ bei ihm angekommen ist, obwohl der Sender ein $A$ übertragen hat. Sensor $\mathrm{s_2}$ steht dann dem Problem

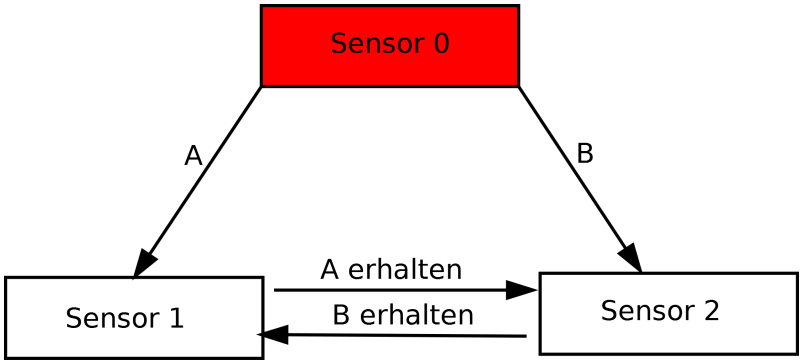



Abbildung 1: Ein 3-Sensornetzwerk mit dem fehlerhaften Sensor 0
Quelle: Eigene Darstellung

gegenüber, zu entscheiden, ob es das Datum $B$ oder $A$ als verwertbar anerkennt. Dieses Beispiel ist unter Abbildung 2 illustriert.

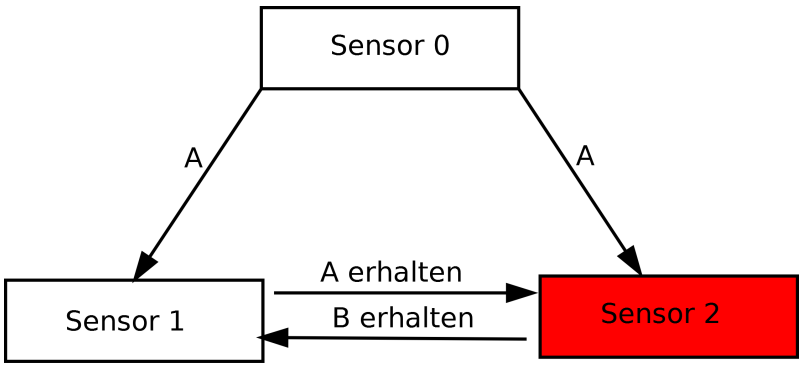



Abbildung 2: Ein 3-Sensornetzwerk mit dem fehlerhaften Sensor 2
Quelle: Eigene Darstellung

Wenn man diesen Fall um einen weiteren ordnungsgemäß funktionierenden Sensor erweitert, so erreicht man einen Zustand, wo der Fehler von einem Sensor toleriert werden kann, d.h. die Fehlfunktion beeinträchtigt in keinster Weise die Systemausführung.
In Abbildung 3 wird diese Erkenntnis mit dem Mastersensor als sogenannter Verräter dargestellt. Für die anderen Konstellationen ist es analog.
Ausgehend von dieser Beobachtung stellt man fest, dass mehr als zwei Drittel des betrachteten Systems aus korrekt funktionierenden Komponenten bestehen muss, um Byzantinische Fehler zu kompensieren. Damit stellt die Anforderung an das System mehr als 3f+1 korrekte Bestandteile vorzuweisen, wobei f die Anzahl an nicht intakten Einheiten charakterisiert, eine weitere Voraussetzung an die Lösung. Man kann zeigen, dass wenn diese Bedingung gegeben ist,dass das System keine Schwierigkeiten besitzt mit f fehlerhaften Sensoren umzugehen.
Der Beweis für die Garantie, dass bei Erfüllung dieser Bedingung implizit die richtige Ausführung des Systems angenommen werden kann, funktioniert über einen Widerspruchsbeweis. Folglich nehmen wir zunächst die negierte Form der Bedingung an. Demnach reichen 3f oder weniger Sensoren, um die Defizite von f Sensoren zu maskieren. Das Beweisverfahren, was auch im Papier *The Byzantine Generals Problem* [LSP82] aufgeführt wird, stützt sich auf die Tatsache, dass der Fall mit drei Sensoren (3-Sensor-Problem), von denen eine fehlerhaft ist, nicht lösbar ist. Ohne Beschränkung der Allgemeinheit wird die Menge 3f in drei gleichgroße Teilmengen aufgeteilt. Jede Menge wird von einem der drei Sensor aus dem 3-Sensor-Problem dargestellt. Da eines dieser Sensoren Mängel beinhaltet und auch genau f Sensoren repräsentiert, bedeutet dies, dass auch genau f Sensoren nicht brauchbare Daten produzieren.
Der Master im 3-Sensor-Problem steht für den Master in der Menge 3f und f-1 weiteren Sensoren. Somit wurde die Problematik mit einem System bestehend aus 3f Sensoren auf das 3-Sensor-Problem reduziert. Da allerdings dieser nicht erfüllbar ist, erhalten wir dasselbe Resultat für die 3f

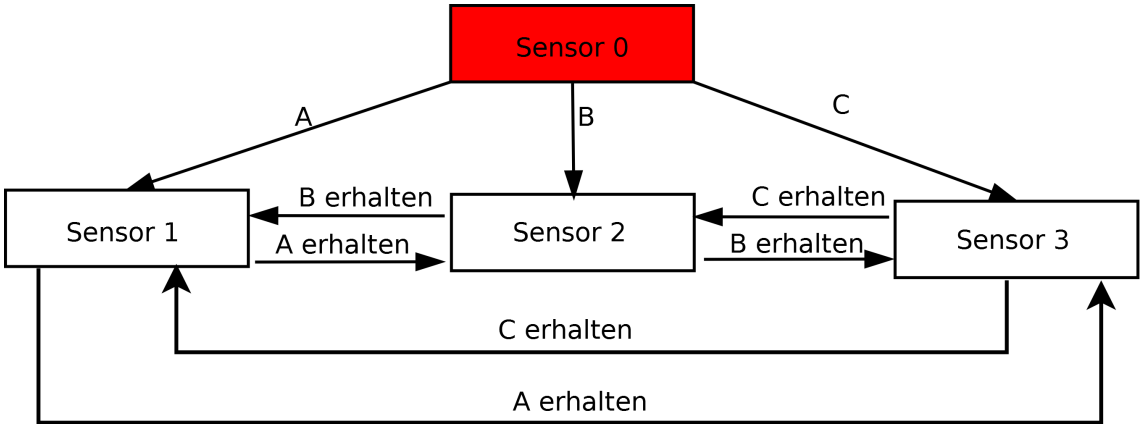



Abbildung 3: Ein 4-Sensornetzwerk mit dem fehlerhaften Sensor 0
Quelle: Eigene Darstellung

Sensoren. Diese Tatsache widerspricht sich jedoch mit der oben aufgestellten Annahme. Demzufolge wurde gezeigt, dass tatsächlich min. 3f+1 Sensoren für die Tolerierung von f nicht korrekt laufenden Sensoren von Nöten sind.
Nun sind alle Voraussetzung für die Lösung des ursprünglichen Problems für den Umgang mit Byzantinischen Mängeln in einem System vollständig.

# 4 Lösung

Ein wichtiges Kriterium für die algorithmische Lösung des Problems ist die Kommunikation zwischen den Sensoren. Daher ist es angebracht an dieser Stelle zunächst alle Anforderungen bezüglich der übermittelten und empfangenen Nachrichten zu erheben.
Als erstes wird angenommen, dass alle gesendeten Nachrichten korrekt übertragen wurden, denn die dabei eventuell entstehenden Schwierigkeiten sind nicht Gegenstand dieser Arbeit. Eine weitere Voraussetzung ist die eindeutige Identifizierung der Nachrichten nach ihren Sendern, was zusammen mit dem ersten Punkt garantiert, dass die fehlerhaften Sensoren mit merkwürdigen Nachrichten unter Angabe anderer Absender-Sensornummern keine anderen Sensoren verwirren können.
Letzteres ist auch möglich zu erfassen, ob ein Sensor eine Nachricht verschickt hat oder nicht, um so effizient nicht korrekt funktionierende Sensor zu erkennen und anstelle dessen ein vorab gewähltes Default-Wert anzunehmen. Es ist hier ausdrücklich zu betonen, dass im Folgenden Algorithmus es nicht darum geht, die fehlerhaften Sensor zu finden und ihre Defizite zu beheben, sondern lediglich eine Koexistenz zu schaffen, wo sowohl richtige als auch falsche Informationen verwaltet werden ohne die korrekte Ausführung des Systems zu gefährden.
Aus den bisher gesammelten Ideen kann man dann induktiv den folgenden Algorithmus, der in Pseudosprache verfasst ist, aufstellen.

Listing 1: Algorithmus OM(0)

```
1 (1) Master sendet seinen Wert v_0 an alle anderen Sensoren;
2
3 (2) Jeder Sensor verwendet den Wert v_0 oder falls dieser nicht vorhanden ist den DEFAULT–
      Wert;
```

Listing 2: Algorithmus OM(m)

```
1 (1) Master sendet seinen Wert v_0 an alle anderen Sensoren;
2
3 (2) Für jedes i gilt:
4             Sei v_i der Wert den Sensor i vom Master erhalten hat (oder DEFAULT–Wert);
```

```
5               Sensor i agiert wie der Master im OM(m-1) um sein Wert v_i den anderen n-2
                    Sensoren zu senden;
6
7   (3) Für jedes i mit j ≠ i gilt:
8               Sei v_j der Wert, den Sensor i von Sensor j in Schritt (2) (OM(m-1))
9               erhalten hat (oder DEFAULT-Wert);
10              Sensor i benutzt die Funktion majority(v_1 ,... ,v_n - 1);
```

Die Funktion *majority* ermittelt erwartungsgemäß die absolute Mehrheit alle erhaltenen Werte. Allerdings ist diese Prozedur nicht zwingend notwendig für die Lösung, sondern kann je nach Wunsch und Bedarf durch eine andere Funktion, die auch über eine Menge definiert ist, verwendet werden.
Die Ausgabe der gewählten Prozedur ist dann die Entscheidung, ob im Anschluss die Sprinkler aktiviert werden oder die Temperaturbeobachtung fortgeführt wird. In Abbildung vier wird ein Beispiel aufgezeichnet, wo Sensor 2 die nicht funktionstüchtige Komponente im System ist. Die falsche Ausgabe von Sensor 2 ist für das System unbedeutend, denn sowohl $s_1$ als auch $s_2$ nehmen das Datum von der Menge ihrer erhaltenen Nachrichten, was am häufigsten vorkommt nämlich A. Aufgrund der Tatsache, dass $s_2$ sich in einem undefinierten Zustand befindet, ist dessen Ausgabe unbekannt. Folglich verursacht das Fehlverhalten von $s_2$ keinen Schaden.
Der korrekte mathematische Beweis von diesem Algorithmus kann der Arbeit von Lamport [LSP82] entnommen werden.

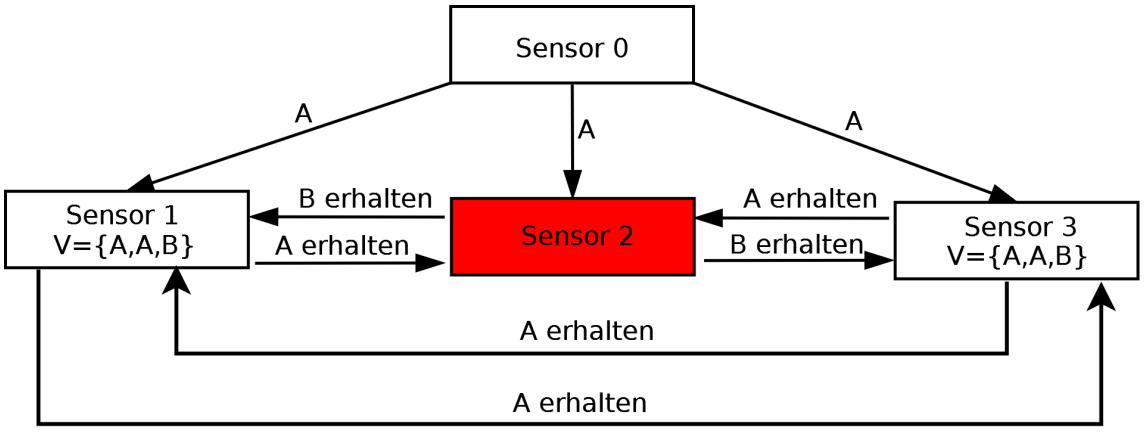



Abbildung 4: OM Algorithmus mit dem fehlerhaftem Sensor 2.
Quelle: Eigene Darstellung

# 5 Erweiterungen und Optimierungen

In diesem Kapitel werden Ansätze für Erweiterungs- und Optimierungsvorschläge vorgestellt. Dabei werden Aspekte wie die Reduzierung der Kommunikationspfade, Senkung der Kosten und die Vereinfachung der Annahme über den Anteil an korrekt funktionierenden Bestandteilen im System näher betrachtet. Grund hierfür ist der allgemeine Wunsch, sowie es in allen wirtschaftlichen Branchen auch üblich ist, Verfahren zu finden, die so wenig wie nur möglich Ressourcen verbrauchen und unnötige Ausgaben vermeiden, denn wie bereits oben dargestellt, werden fehlertolerante Systeme im Bezug auf Byzantinische Fehler selten implementiert wegen den relativ hohen Kosten.

Die zentrale Idee im unter Kapitel vier repräsentierten Algorithmus OM ist die Versendung des persönlichen Wertes $v_i$ von Sensor $i$ an alle anderen Sensoren.
Aufgrund der begrenzten Anzahl an sogenannten Verrätern ist garantiert, dass zwischen allen Knoten bzw. Systemkomponenten ein direkter Kommunikationspfad besteht. Jedoch ist dieses Kriterium bei der Konstruktion des Systems kostenverursachend und daher zu umgehen, denn mit jedem zusätzlichen Pfad wachsen die Gesamtausgaben stark an. Daher besteht der Bedarf nach einer nicht eng verstrickten Netzwerkstruktur, der dennoch alle bisher aufgestellten Eigenschaften nach wie vor beinhaltet.
Graphentheoretisch betrachtet waren die Sensoren bisher unter einem vollständig ungerichteten Graphen mit Mehrfachkanten zusammengefasst. In Abbildung 5 ist solch ein Graph bestehend aus sechs Sensoren und 15 Verbindungen gemäß der bisher vorgestellten Lösung des Byzantinischen Problems illustriert.

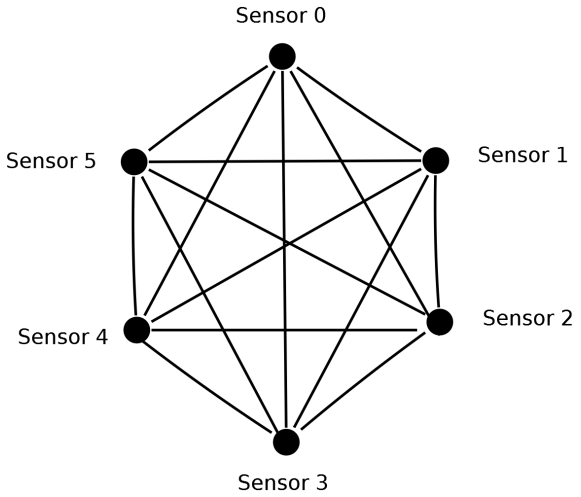



Abbildung 5: vollständiger Graph mit sechs Knoten
Quelle: Eigene Darstellung

Nun geht man von einem ungerichtetem Graphen aus, der p-regulär ist, d.h. alle Knoten in diesem Graphen haben eine reguläre Menge von Nachbarn. Damit wird verdeutlicht, dass alle Knoten zu p anderen Knoten eine direkte Verbindung haben und alle anderen Knoten über Pfade mittels ihrer Nachbarknoten erreichen können.
Dementsprechend besteht die Netzwerkstruktur wie gewünscht aus der minimalen Anzahl an Verbindungen ohne die Voraussetzungen zu verletzen. Dahingehend spiegelt Abbildung 6 diese Modifikation, was in Abbildung 5 bisher nicht berücksichtigt war.

Diese Erkenntnis muss auch in die ursprüngliche Lösung mit integriert werden. Das Grundkonzept des Algorithmus bleibt jedoch nach wie vor erhalten. Der einzige bedeutende Unterschied ist, dass wie geplant man nicht allen Sensoren direkt Daten übertragen kann, sondern die Sensoren, die keine direkten Nachbarn sind über Pfade, die mittels der tatsächlichen Nachbarknoten gebildet werden, zu erreichen und so die Informationen zu verbreiten. Folglich bleibt der Algorithmus von der Idee her gleich, aber seine Umsetzung wird eine wenig modifiziert.

Listing 3: Algorithmus OM(m p)

```
1  (0) Wähle eine Menge N bestehend aus p Nachbarn vom Mastersensor;
2
3  (1) Dem Sensor i ∈ N werden die Daten vom Master direkt übermittelt;
4  (2) Für jedes i ∈ N gilt:
5              Sei v_i der Wert vom Sensor i, den er vom Master
6              erhalten hat (oder DEFAULT-Wert);
7              Sensor i sendet v_i an allen anderen Sensoren k wie folgt:
8                              (A) if (m=1) Sensor i sendet den Wert über Pfad γ_i,k;
9                              (B) if (m>1) Sensor i agiert wie der Master in OM(m-1,p-1)
10                                          mit dem neuen Graph ohne den Master;
11
12 (3) Für jedes k und jedes i mit i,k ∈ N , i ≠ k gilt:
```

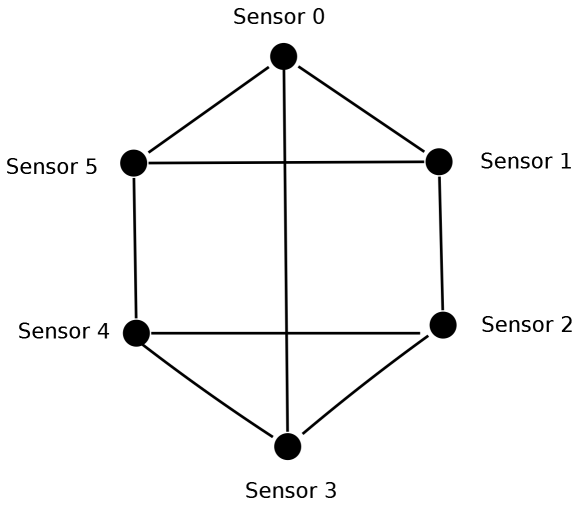


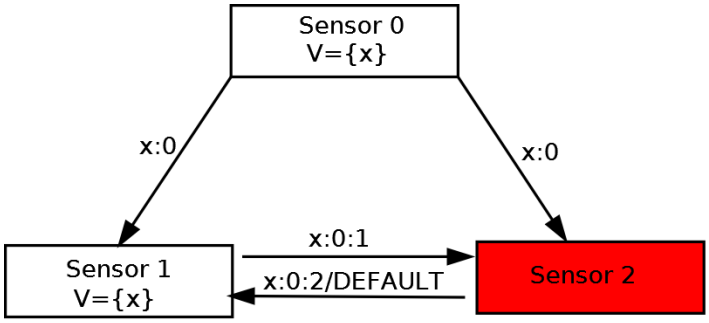

Abbildung 6: Ein 3-regulärer Graph mit sechs Knoten
Quelle: Eigene Darstellung

```
13          Sei v_i der Wert, den Sensor k vom Sensor i in Schritt (2) (OM(m-1))
14          erhalten hat (oder DEFAULT-Wert);
15          Sensor i benutzt die Funktion majority(v_1 ,... ,v_n - i);
```

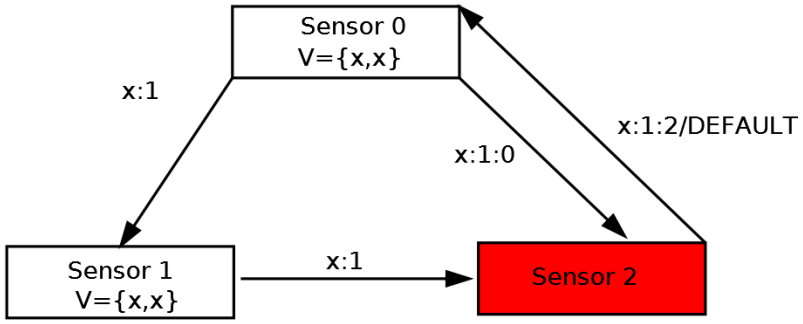
Trotz der effizienten Umgestaltung des Programms gibt es dennoch wirksamere Lösungen, auf denen sogar Lamport in seiner Arbeit verweist [Dol82].
Die Forschung entwickelt sich hinsichtlich der Byzantinischen Fehler dahingehend, dass man versucht die Kosten, die in der algorithmische Lösung von Lamport relativ hoch sind, stark einzugrenzen. Neben Dolev [Dol82] gibt es noch weitere Autoren, die an diesem Ziel arbeiten. In der Arbeit *u*nter [KAD+09] wird zum Beispiel eine Möglichkeit gezeigt, wie man die Leistung des Systems trotz eines eingebauten Protokolls für die Tolerierung von Byzantinischen Fehlern relativ hoch positionieren kann. Das Protokoll wird in der Publikation mittels eines Servernetzwerkes beschrieben. Dieses Netzwerk dient ausschließlich der Kompensierung von potentiellen Fehlern, d.h. neben dem Masterserver gibt es noch weitere Server, die dieselbe Aufgaben erfüllen und dementsprechend redundant sind. Dabei werden nämlich die Anfragen eines Klienten unmittelbar bevor die Lösungen zwischen den Sensoren ausgetauscht wird, um eventuelle Fehler zu entdecken, beantwortet, und zwar von allen Servern. Bei der Präsenz eines Byzantinischen Fehlers werden dann folglich dem Klienten nicht identische Nachrichten gesendet, der dieses merkwürdige Verhalten zu Kenntnis nimmt und solange wartet, bis die Informationen und damit das gesamte System wieder einen stabilen Zustand erreicht. Die Server registrieren dann durch die unbeantworteten Nachrichten das Systemversagen und leiten anschließend die hierfür erforderlichen Gegenmaßnahmen ein. Ab dem Schritt kommt die bekannte Lösung zum Einsatz. Folglich werden hier keine vorbeugenden Maßnahmen getroffen, sondern erst bei Auftritt des Fehlers reagiert, was die Leistungsfähigkeit enorm fördert.
Im letzten Teil dieses Kapitels wird eine weitere Erweiterungsalternative präsentiert, die sich mit der Problematik beschäftigt, den Toleranzbereich der bisherigen Lösung um ein Mehrfaches zu erhöhen.
Bisher wurde stets die Voraussetzung bei f fehlerhaften Sensoren 3f+1 Sensoren einzusetzen, für die korrekte Lösung angenommen. Entsteht allerdings der Bedarf eine höhere Anzahl an funktionsuntüchtigen Bestandteilen zu tolerieren, dann kann dies mittels Kryptographie relativ simpel gelöst werden. Kann man nämlich über eine zusätzliche externe Anwendung die Authentifizierung der Daten garantieren, dann ist sogar eine Lösung mit beliebig vielen nicht korrekt funktionieren Sensoren laut Lamport [LSP82] realisierbar.
Hierfür wird jedem Sensor eine Menge $V_i$ zu Verfügung gestellt, der wie bisher die ankommenden Nachrichten $v_j$ speichert und ist erneut für die Wahl des richtigen Datums ausschlaggebend.

Zusätzlich werden die erhaltenen Daten mit einer Signatur gekennzeichnet und an alle weiteren Sensoren übermittelt.
Unter der Annahme, dass die Signatur der anderen Netzwerkteilnehmer eindeutig zuordbar ist und auch eine Verfälschung ausgeschlossen werden kann, wird der Einfluss an fehlerhaften Sensoren vollkommen eliminiert.
Sei $v : i : j$ die Nachricht, den Sensor $i$ signiert dem Sensor $j$ übermittelt hat und dieser anschließend seine eigene Signatur hinzufügt, dann ist nun die Aufgabe vom Sensor $j$ die Nachricht $v : i : j$ an die restlichen Sensoren zu übertragen und v in die Menge $\mathrm{V_j}$ aufzunehmen. Demzufolge besitzt jeder Sensor $i$ nach Beenden der Sendephase eine Menge von Nachrichten der Form $v : 0 :\mathrm{j_1}: ... :\mathrm{j_k}:i$ und kann nun auf Grundlage von $\mathrm{V_i}$ eine Entscheidung hinsichtlich der künftig zu verwendeten Informationen zu treffen.
Die Authentifizierung, die zum Beispiel durch Publik-Key-Kryptographie realisiert werden kann, garantiert hier, dass kein Knoten die Möglichkeit besitzt die angekommenen Daten verfälscht weiterzuleiten, sondern nur eigene persönliche Daten mit falschen Inhalten ausgestattet, zu versenden. Mit anderen Worten kann man mit dieser Lösung die Informationen, die ein Knoten versendet hat, diesen Knoten eindeutig zuordnen und kann mithilfe dieser Eigenschaft all diejenigen falschen Komponenten ausschließen, die durch die Manipulation der erhaltenen Daten oder aufgrund einer Fehlfunktion produzierten verfälschten Informationen die Ausführung des Gesamtsystems in Gefahr bringen. Der genaue Algorithmus mit dem entsprechenden Korrektheitsbeweis ist im Papier unter [LSP82] vorgeführt.

Das unter Abbildung 7 präsentierte Beispiel zeigt zwei Zustände des 3-Sensor-Problems, wo jeweils die beiden funktionstüchtigen Sensoren ihre Daten an alle versenden. Aus den vorhergehenden Kapiteln war bekannt, dass dieser Byzantinische Fehler nicht ohne Auswirkungen auf das System tolerierbar war. Verwendet man jedoch Signaturen wie in dem oben beschriebenen Verfahren, so ist diese Problematik lösbar.
Im ersten Schritt wird der tatsächliche Wert, der vom Mastersensor mit der ID 0 berechnet wurde, ermittelt. Sensor 0 hat in diesem Beispiel das Ergebnis $x$ für die Berechnung $k$ herausgefunden und teilt dies den beiden anderen Sensoren mit. Der Sensor $\mathrm{s_1}$, der die Nachricht x:0 erhalten hat, signiert diesen und sendet es den Sensor $\mathrm{s_2}$. Sensor $\mathrm{s_2}$ hingegen ist nicht dazu in der Lage die Nachricht x:0 zu verfälschen und sendet entweder die Nachricht x:0:2 korrekt an $\mathrm{s_1}$ oder führt keine Aktion aus. Folglich registriert $\mathrm{s_1}$, dass $\mathrm{s_0}$ für die Aufgabe $k$ das Ergebnis $x$ berechnet hat. Der darauffolgende Schritt, wo es darum geht dasselbe Resultat $x$ von $\mathrm{s_1}$ für $k$ den anderen Sensoren mitzuteilen, ist analog.
Im dritten Schritt, was in Abbildung 8 dargestellt sit, erhalten beide $\mathrm{s_0}$ und $\mathrm{s_1}$ entweder dieselbe Nachricht y:2 mit dem falsch berechneten Ergebnis von $\mathrm{s_2}$ oder sie erhalten beide entgegengesetzte Werte. Für diesen Fall, wo $\mathrm{s_0}$ mit der Nachricht $z : 2$ und $\mathrm{s_1}$ mit die Nachricht $y : 2$ umgehen müssen, wird sofort erkannt, dass $\mathrm{s_2}$ fehlerhaft ist, da $\mathrm{s_0}$ und $\mathrm{s_1}$ sich bezüglich der erhaltenen Daten austauschen. Anderenfalls verrät die Menge $\mathrm{V_0}$ bzw. $\mathrm{V_1}$, dass das Resultat für die Berechnung $k$ $x$ beträgt.
Das 3-Sensor-Problem ist folglich mittels Kryptographie lösbar und zeigt, dass die algorithmische Lösung mit Signaturen einen größeren Toleranzbereich für Fehler zulässt und damit eine ernsthafte Alternative zu der ursprünglichen Lösung ist.
Trotz dieser vorteilhaften Eigenschaft wird das Verfahren jedoch sehr selten eingesetzt, weil der hierbei entstandener zusätzlicher Aufwand für die Authentifizierung für zum Beispiel das Sensornetzwerk unter Kapitel eins, aber auch für viele andere Anwendungen bedenklich ist. Daher ist es nur für sehr anfällige und sicherheitskritische Systeme zu empfehlen.




Schritt 1: Sensor 0 sendet x

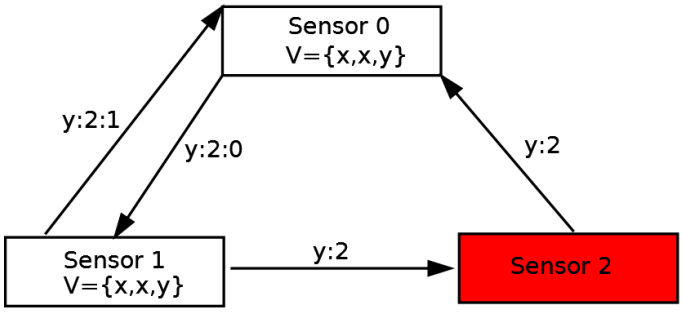



Schritt 2: Sensor 1 sendet x

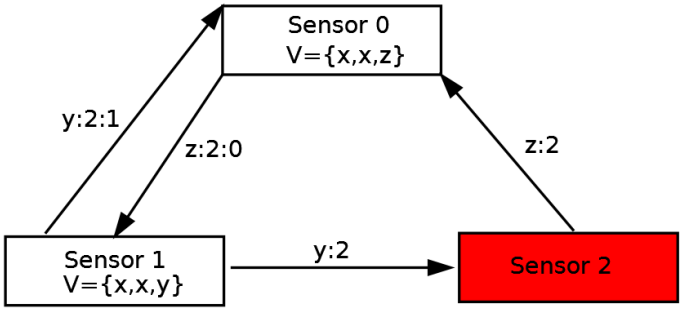
Abbildung 7: Verfahren mit Signatur: Sensor 2 ist fehlerhaft
Quelle: Eigene Darstellung

# 6 Fazit

Die Rolle von Byzantinische Fehlern in den aktuellen Systemen wurde mittels dem Sensornetzwerk für eine Feuermeldeanlage gezeigt. Außerdem wurden verschiedene Lösungs- und Optimierungsalternativen und die hierfür erforderlichen Voraussetzungen vorgestellt.
Wenngleich einige Fortschritte in der Forschung, die im Kapitel fünf dargestellt sind, erzielt wurden, in denen auch kostengünstigere und besonders sichere Erweiterungen der ursprünglichen Lösung von Lamport vorgeschlagen wurde, gibt es dennoch zahlreiche Entwickler von verteilten Systemen, die sich gegen eine Fehlertoleranz hinsichtlich byzantinischer Fehler entscheiden [DHSZ]. Grund dieser äußerst konservativen Haltung ist wie bereits erwähnt die relativ hohen Ansprüche der Lösung bezüglich des Aufwands für die Implementierung und der zusätzlichen Kosten verursacht durch die ununterbrochene Versendung von Nachrichten zwischen den Bestandteilen, was in der Tat für Echtzeitsysteme eine Hürde darstellt. Allerdings gibt es bereits viele Hilfestellungen bei der Implementierung solcher sicherer Systeme. In der Publikationen [CL02] wird zum Beispiel ein Algorithmus vorschlagen, mit denen man sogar Byzantinische Fehler reproduzieren kann, was bei der Entwicklung eines fehlertoleranten Systems von Nutzen sein kann.
Außerdem wurde mithilfe des unter Kapitel 2 analysierten Anwendung verdeutlicht, dass diese Fehler in kritischen Systemen nicht nur einen hohen finanziellen Schaden anrichten, indem sie die korrekte Ausführung des Systems negativ beeinflussen, sondern auch menschliches Leben gefährden können. Die Feuermeldeanlage ist ein Beispiel von vielen Systemen, wo die Konsequenzen nicht verantwortbar sind.
Beispielsweise können die im Verkehrsnetz eingesetzten Sensoren genauso von dieser Fehlerklasse betroffen sein. Das in diesem System entstandene Byzantinische Fehler stören die Koordination des Verkehrsflusses und können daher wieder eine Gefahr für Menschen darstellen. Dies ist ein Beispiel von vielen sicherheitsskritischen Systemen deren Zuverlässigkeit und Robustheit von größter Bedeutung sind. Daher wird die Verantwortung der Entwickler solcher Systeme an dieser




Schritt 3a: Sensor 2 sendet y




Schritt 3b: Sensor 2 sendet y und z

Abbildung 8: Verfahren mit Signatur: Sensor 2 ist fehlerhaft
Quelle: Eigene Darstellung

Stelle besonders betont und geraten dies im Blick zu behalten

## Literatur

[910] *Byzantine fault tolerance*. : *Byzantine fault tolerance*, Juni 2010. – `http://en.wikipedia.org/wiki/Byzantine_fault_tolerance` (Abrufdatum: 14.06.20010)

[CL02] CASTRO, M. ; LISHOV, B.: Practical Byzantine Fault Tolerance and Proactive Recovery ACM Transactions on Programming Languages and Systems vol. 20, no. 4, 2002, S. 398–401

[DHSZ] DRISCOLL, Kevin ; HALL, Brendan ; SIVENCRONA, Hakan ; ZUMSTEG, Phil: Byzantine Fault Tolerance, From Theory To Reality, vol. ume 2788 / 2003. In: *Computer Safety, Reliability, and Security* Bd. 2788/2003. Springler-Verlag. – ISBN 3–540–20126–2, S. 235–248

[Dol82] DOLEV, D.: The Byzantine Generals Strike Again / Stanford University. 1982. – Journal of Algorithm, vol. 3. – 14–30 S.

[KAD+09] KOTLA, R. ; ALVIS, L. ; DAHLIN, M. ; CLEMENT, A. ; WONG, E.: Zyzzyva: Speculative Byzantine Fault Tolerance Proc. 21st ACM Symp. Operating Systems Principles (SOSP 07), 2009

[LSP82] LAMPORT, L. ; SHOSTAK, R. ; PEASE, M.: The Byzantine Generals Problem ACM Transactions on Programming Languages and Systems,vol. 3, 1982, S. 382–401