Verteilte Systeme
Übung 3

Tillmann Faust

29.9.2024

Zusammenfassung

Dieser Abschlussbericht ist im Rahmen der Veranstaltung "Verteilte Systemeän der Universität Trier im Sommersemester 2024 entstanden und stellt das Paxos Agreement Protokoll und eine simple Implementierung dessen in Java vor.

Inhaltsverzeichnis

1 Einleitung

Das Gebiet der verteilten Systeme in der Informatik beschäftigt sich mit Gruppen unabhängiger Systeme, welche durch ein Netzwerk miteinander verbunden sind. Diese Systeme haben die Aufgabe gemeinsam eine oder mehrere Aufgaben zu lösen, welche von der Kommunikation und Weiterleitung von Informationen, über die Verbesserung der Verfügbarkeit von Diensten, bis hin zur Aufteilung von Rechen- oder Speicherleistung reichen. Dabei müssen verteilte Probleme eine Reihe von Problemen lösen, welche ansonsten die Leistung des Systems einschränken oder ganz verhindern können:

- Fehlertoleranz
- Skalierbarkeit
- Konsistenz und Verfügbarkeit
- Synchronisation
- Latenz

Ein weiteres zentrales Problem ist die Konsensfindung, bei welcher sich das verteilte System, auch unter dem Auftreten von Fehlern, auf gemeinsame Werte oder Zustände einigen können muss. Auf ein Protokoll zum behandeln dieses Problems, Paxos, soll im weiteren Verlauf genauer eingegangen werden.

2 Paxos

2.1 Konsens in verteilten Systemen

Das Konsensproblem tritt dann auf, wenn sich ein verteiltes Netzwerk auf einen gemeinsamen Wert oder Zustand einigen soll. Da es in Netzwerken immer wieder zu Fehlern durch ausfallende oder fehlerhafte Knoten, oder Latenz und Timing Probleme kommen kann bedarf es robuster Protokolle um eine Korrekte Entscheidungsfindung dennoch zu ermöglichen.

2.2 Historischer Hintergrund

In diesem Zusammenhang schrieb Leslie Lamport 1989 die Arbeit ”The Part-Time Parliament” [1], zuerst im Versuch zu beweisen, dass es keinen korrekten Algorithmus geben kann. Dieses Paper wurde anschließend beinahe 10 Jahre in keiner Zeitschrift akzeptiert, da es für die meisten Leser zu undurchsichtig war und wurde letzendlich erst 1998 veröffentlicht. Daraufhin veröffentlichte Lamport 2001 eine zweite Arbeit "Paxos Made Simple"[2], deren Zusammenfassung liest: ”Paxos, when presented in plain english, is very simple”.

2.3 Funktionsweise

Im Kern besteht ein Paxos Netwerk aus drei Arten Von Knoten: Proposern, Acceptors und Learners, Knoten in einem Paxos Netzwerk können eine oder mehrere dieser, und weiterer, später genannten, Rollen einnehmen. Proposer können Werte Vorschlagen, welche vom System akzeptiert werden sollen, Acceptors entscheiden darüber, welcher Wert akzeptiert werden soll und Learner sollen über

die Prozesse, speziell die Ergebnisse des Entscheidungsprozesses informiert werden. Wichtig dabei ist, dass ein Wert erst dann als gewählt verkündet wird, wenn er tatsächlich gewählt wurde. Genauer besteht Ablauf eines Agreement-Prozesses darin, dass ein oder Mehrere Proposer einen Wert vorschlagen wollen. Dazu wählt jeder Proposer eine einzigartige Proposal Nummer, das heißt eine Zahl, welche bisher noch von keinem anderen Proposer verwendet wurde und sendet diese zusammen mit einem Uniquifier, z.B. der Proposer-ID an ein Quorum von Akzeptoren. Zusätzlich schreibt der Proposer sowohl die Proposal Nummer, als auch der Wert in den Persistenten Speicher.

Definition 1 (Quorum)**.** *Ein Quorum ist eine Gruppe von mindestens so vielen Acceptors, dass sich zwei Quoren immer mindestens einen Acceptor teilen. Jedes Quorum umfasst also mindestens mehr als die Hälfte aller Acceptors.*

Die Acceptors entscheiden anschließend darüber, ob sie bereits eine höhere Proposal Nummer erhalten haben, oder ob die erhaltene die höchste bisher ist. Wenn zuvor keine Proposal Nummer erhalten wurde muss die erste genommen werden, da sonst kein Fortschritt im System garantiert werden kann. Wird die vorgeschlagene Nummer akzeptiert, so sendet der Acceptor ein Promise mit der zugehörigen Nummer zurück und schreibt diese in den persistenten Speicher, andernfalls wird ein Nack gesendet.

Definition 2 (Promise)**.** *Ein Promise ist die Zusage, bzw. der Garant eines Acceptors, keine Werte mit niedrigeren Nummern, als die erhaltene Proposal Nummer, zu akzeptieren.*

Hat ein Acceptor bereits einen Wert von einem anderen Proposer akzeptiert, so sendet er, als Teil seines Promises, die vorherige Nummer und den zugehörigen Wert. Wenn ein Proposer ein oder mehrere solcher Nummern und Werte erhält, wählt er den Wert mit der höchsten Nummer aus und vertritt diesen fortan. Sobald ein Proposer Promises von mindestens mehr als der Hälfte an Acceptors erhält geht er in die nächste Phase über, in welcher er an jeden Acceptor im Quorum ein Accept, bestehend aus der Proposal Nummer und dem aktuell vertretenen Wert, sendet. Die Acceptors können diese Nummer nun mit der aktuell gespeicherten Nummer vergleichen. Sind die Nummern gleich, wird der gesendete Wert in den persistenten Speicher geschrieben und ein 'accepted' zurückgesendet, ansonsten wird ein Nack gesendet. Sobald der Proposer von einer Mehrzahl der Acceptors ein 'accepted' erhalten hat, werden die Learner und alle anderen Knoten im Netzwerk über die erfolgreiche Wahl informiert. Wenn eine längere Zeit nichts im Netzwerk passiert, so kann jederzeit ein Proposer eine neue Propose-Nachricht an ein Quorum senden, welche entsprechend des obigen Verlaufs behandelt wird.

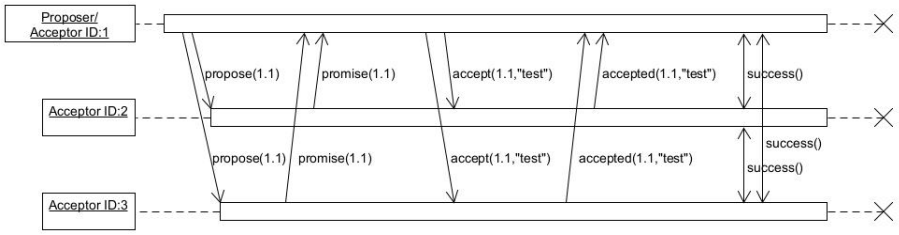

Abbildung 1: Sequenzdiagramm eines erfolgreichen Durchlaufs

2.3.1 Fehlerbehandlung

Agreement Protokolle müssen in der Lage sein mit verschiedenen Arten von Fehlern umgehen zu können.

Timing und Latenz Während eines Durchlaufes von Paxos fahren Proposer nur zur nächsten Phase fort, wenn die vorherige Phase als erfolgreich abgeschlossen gilt. Dabei wird, zur Not auch eine Lange Zeit, auf jeden fehlenden Wert gewartet.

Nachrichtenverlust Beim Verlust von Nachrichten kann es passieren, dass entweder keine oder nicht genügend Proposal, Accept oder Accepted Nachrichten ankommen, in dem Fall wird irgendwann ein Proposer ein neues Proposal stellen. In diesem Fall wird der Wert dieses, oder eines anderen späteren Proposals akzeptiert, oder ein bereits teilweise akzeptierter Wert wird von diesem Proposer weiter propagiert.

Ausfall Bei Paxos kann es an zwei wesentlichen Stellen zu Ausfällen kommen: Bei den Proposern und bei den Acceptors. Ein Paxos Netzwerk mit $2n + 1$ Acceptors ist dabei in der Lage den Ausfall von n zu kompensieren. Abbildung 2 Zeigt ein Netzwerk mit einem Proposer/Acceptor und zwei Acceptoren, von denen einer ausfällt.

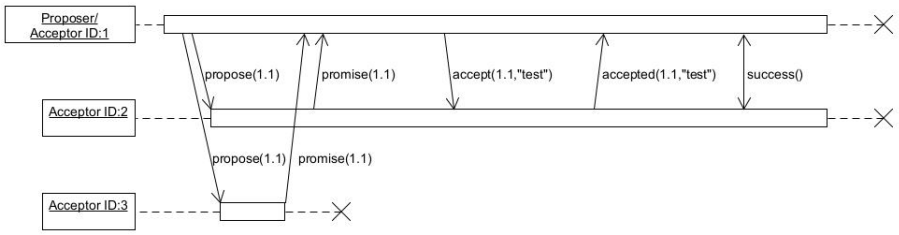

Abbildung 2: Sequenzdiagramm eines Acceptor Ausfalls

Weiterhin können Proposer an vier wesentlichen Stellen ausfallen. Fällt ein Proposer vor dem Senden des ersten Proposal aus, so ist es, als wäre von dem Proposer nie ein Proposal gemacht worden. Fällt der Proposer währen des Proposal Prozesses oder vor dem Senden des ersten Accepts aus, so wird irgendwann

ein neuer Proposer eine neue Nummer vorschlagen, welche die alte ersetzt. Im Letzten Fall fällt der Proposer während dem Senden der Accept Nachrichten aus. In diesem Fall wird ein neuer Proposer ebenfalls einen neuen Vorschlag machen, erfahren, dass ein oder mehrere Werte bereits teilweise akzeptiert wurden und den höchstnummerierten dieser Werte bis zur erfolgreichen Akzeptanz, oder zum eigenen Ausfall, vertreten.

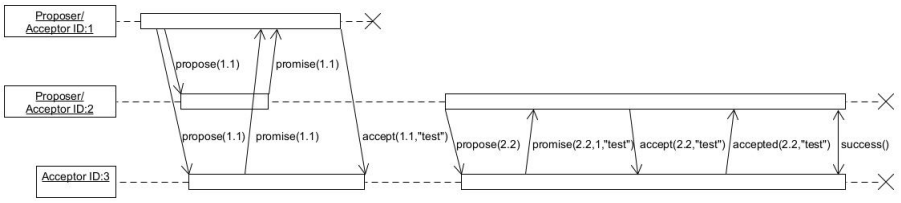

Abbildung 3: Sequenzdiagramm eines Ausfalls während des Accept-Prozesses

2.4 Varianten

2.4.1 Multi-Paxos

Da Paxos sich nur mit der Wahl eines einzelnen Wertes beschäftigt reicht es in vielen verteilten Anwendungen nicht aus. Dazu schlägt Lamport vor, Paxos zu Multi-Paxos zu erweitern, welches in der Lage ist, die Werte, auf die sich in vorherigen Durchläufen geeinigt hat zu speichern und damit eine Folge von akzeptierten Werten zu verwalten. Lamport selbst geht an einigen Stellen nicht detaillierter auf Implementierungsmöglichkeiten ein, weswegen Multi-Paxos in realen System auf viele verschiedene Arten umgesetzt wird

2.4.2 Weitere Optimierungen und Erweiterungen

Das Paxos Protokoll kann um weitere Rollen und Funktionen erweitert werden.

Leader Um den Fortschritt in einem Agreement-Prozess garantieren zu können, kann einem Paxos Netzwerk die Rolle des Leader hinzugefügt werden. Der Leader ist ein spezieller Proposer, welcher alleine Vorschläge macht. Sollte dieser Proposer ausfallen wählt das System einfach einen neuen.

Client Clients sind Knoten, welche von außerhalb mit dem Paxos Netzwerk interagieren um Anfragen an das System stellen. Primär fragen Clients Informationen über den Netzwerkzustand ab oder schlagen selber mögliche Werte vor, welche dann potentiell in den Agreement-Prozess eingebracht werden.

3 Implementierung

Das Paxos Protokoll wurde auf der Basis von sim4da umgesetzt. Die Implementierung besteht aus drei wesentlichen Gruppen, dem PaxosNetwork, den PaxosNodes und Clients und den Messages.

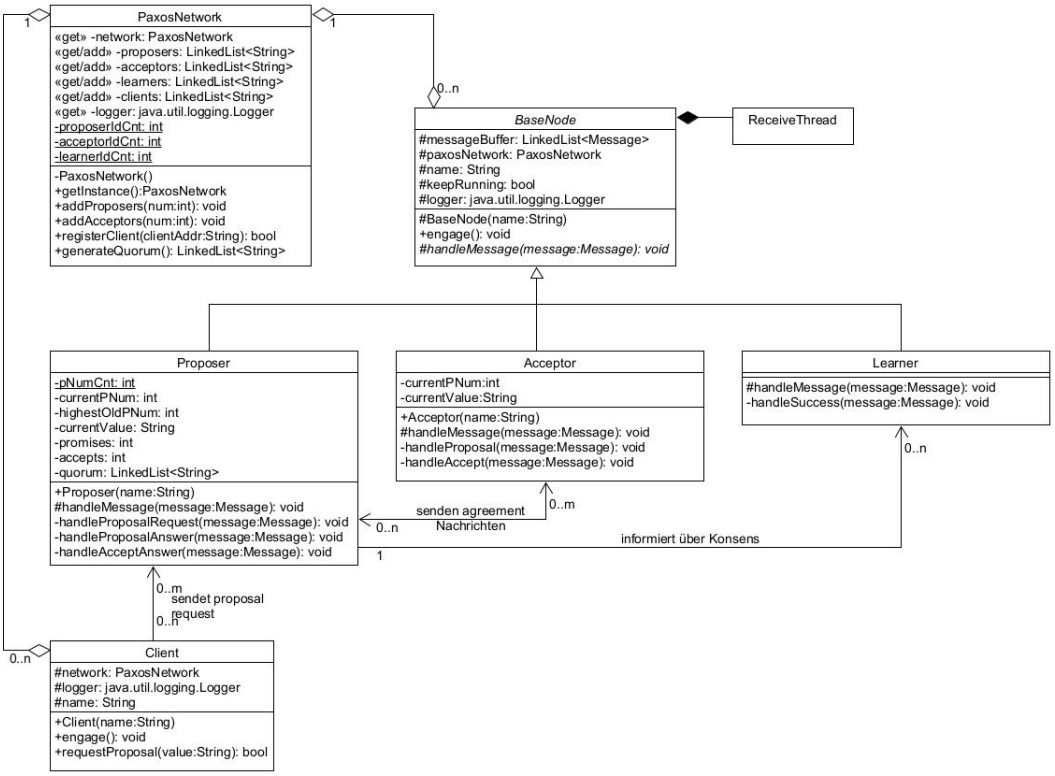

Abbildung 4: Klassendiagramm der Java Implementierung

Zusätzlich zu den in Abbildung 4 gezeigten Beziehungen der Klassen sind BaseNode und Client weiterhin Unterklassen der Klasse org.oxoo2a.sim4da.Node.

3.1 PaxosNetwork

Die Klasse PaxosNetwork dient als Verwalter und "Koordinator" der einzelnen Akteure im Paxos Protokoll. Es weicht insoweit von einer realistischeren Implementierung ab, als dass es jederzeit die Adressen aller Knoten im Netzwerk kennt und jeder Knoten die Möglichkeit hat die Adresslisten der anderen Knoten direkt abzurufen. PaxosNetwork ist als Singleton implementiert, was bedeutet, dass jederzeit nur ein Paxos-Netzwerk gleichzeitig simuliert werden kann. Jede BaseNode verfügt über eine Referenz auf die PaxosNetwork Instanz um den direkten Zugriff zu ermöglichen. PaxosNetwork übernimmt zudem die Aufgabe der Bildung eines Quorums. Dabei hat jedes Quorum genau so viele Acceptors, dass es gerade mehr als die Hälfte sind.

3.2 Messages

Die Akteure im Paxos Netzwerk kommunizieren über eine Reihe von verschiedenen Nachrichten. Jede Nachricht besteht aus einem Header und der Payload, welche aus, für den korrekten relevanten, Informationen bestehen. Der Header jeder Message besteht aus dem Namen (also der Adresse) des sendenden Knoten

und einem ”type” Wert, der angibt um was für eine Nachricht es sich handelt. Da der Name Teil jeden Headers ist wird er im weiteren Verlauf weggelassen.

Proposal Request Die Proposal Request wird von Clients an Proposers versendet um ein Proposal zu erbitten.

Header	
”type”	”proposal_req”
Payload	
”value”	Proposal Wert

Proposal Request Antwort Diese Nachricht dient als Antwort auf Proposal Requests und ist entweder ein Ack oder ein Nack.

Header	
”type”	”proposal_req_ans”
Payload	
”ack”	”ack”/”nack”

Proposal Das Proposal ist die Bitte eines Proposers um die Akzeptanz einer Proposal Nummer an ein Quorums.

Header	
”type”	”proposal”
Payload	
”pNum”	Proposal Nummer

Proposal Antwort Die Antwort eines Acceptors auf ein Proposal. Ein ”nack” bedeutet eine Ablehnung. Wird das Proposal akzeptiert, so wird als Promise ein ”ack”, die neue pNum, sowie die vorherige pNum und, falls vorhanden, ihr zugehöriger Wert zurückgesendet. Gilt beim Acceptor ’currentValue == null’ so werden ”pNumOld” und ”value” vom Proposer ignoriert.

Header	
”type”	”proposal_ans”
Payload	
”ack”	”nack”

Header	
”type”	”proposal_ans”
Payload	
”ack”	”ack”
”pNum”	neue pNum
”pNumOld”	vorherige pNum
”value”	aktuell gespeicherter Wert.

Accept Die Accept-Anfrage eines Proposers an die Acceptors, sobald jeder ein Promise gegeben hat.

Header	
”type”	”accept”
Payload	
”pNum”	Proposal Nummer
”value”	zu akzeptierender Wert

Accept Antwort Antwort eines Acceptors auf eine Accept Nachricht. Besteht aus einem Ack oder Nack, sowie der aktuellen pNum und value.

Header	
”type”	”accept_ans”
Payload	
”ack”	”ack”/”nack”
”pNum”	aktuelle Proposal Nummer
”value”	aktueller Wert

Success Wird vom Proposer, für dessen Wert sich das Netzwerk entschieden hat, an alle Learner gesendet und informiert über die erfolgreiche Wahl.

Header	
”type”	”success”
Payload	
”pNum”	gewählte Proposal Nummer
”value”	gewählter Wert

3.3 Paxos Knoten

3.3.1 BaseNode

Der Einfachheit halber wurden die einzelnen Akteure des Paxos-Algorithmus als separate Klassen implementiert. Um das Projekt simpler zu halten wurde davon abgesehen ein Leader-System zu implementieren.

Proposer Der Proposer besteht aus einer Reihe von Variablen, um den aktuellen verlauf des Agreement-Verfahren akkurat verfolgen zu können.

Variable	Rolle
pNumCnt	Zähler für eindeutige Proposal Nummern. Da keine Thread-Sicherheit eingebaut ist, kann nicht garantiert werden, dass die Nummern tatsächlich 100% einzigartig sind.
currentPNum	Die Proposal Nummer, die aktuell vertreten wird. -1 bedeutet, dass der Proposer noch keine Nummer besitzt
highestOldPNum	Speichert in Fällen, in denen ein oder mehrere Acceptors bereits ein Accept gegeben haben, die höchste bisher gesehene Proposal Nummer
currentValue	Speichert den aktuell vertretenen Wert.
promises	Zählt die erhaltenen Promises
accepts	Zählt die erhaltenen Accepts

Tabelle 1: Variablen der Proposer-Klasse und ihre Rolle

Ein regulärer Programmabluf sieht so aus, dass ein Proposer zuerst eine proposalRequest erhält, wenn noch kein Wert promoted wird, wird der vom Client gewünschte Wert genommen. Anschließend sendet der Proposer ein Proposal an ein Quorum an Acceptors und zählt anschließend jedes Promise. Erhält der Proposer in diesem Zeitraum irgendein Nack, so wird er niemals auf die nötige Zahl an Promises kommen und friert effektiv ein. Erfährt der Proposer in diesem Schritt, dass bereits ein Wert von einem Acceptor angenommen wurde, so wird er den höchstwertigen erhaltenen Wert vertreten. Erhält der Proposer von jedem Acceptor ein promise, so wird eine Accept Request an jeden Acceptor geschickt. Erhält er auch hier ein Accept von jedem Acceptor so wird eine Success Nachricht an jeden Learner im Netzwerk gesendet. Ansonsten friert der Proposer effektiv ein.

Acceptor Einfacher, als der Proposer, hat ein Acceptor lediglich die Aufgabe eingehende Propoals zu bewerten und entweder ein Promise zurückzugeben, falls die erhaltene Proposal Number die bisher höchste ist oder abzulehnen, falls nicht. Wenn der Acceptor zuvor bereits ein Accept gesendet hat, wird der Proposer zusätzlich über die Nummer und den Wert des Accepts informiert. Gleichfalls antwortet der Acceptor auf Accept-Request-Messages entweder mit einem Accept, wenn die Nummer weiterhin dieselbe ist, oder einem Nack, wenn in der Zwischenzeit eine höhere Proposal Nummer erhalten wurde.

Learner Der Learner wird vom Proposer informiert, sobald dieser ein Accept von jedem Knoten seines Quorums erhalten hat, d.h. wenn das Netzwerk sich auf einen Wert geeinigt hat. In diesem Fall wird das Ergebnis des Agreement-Protokolls gelogged.

3.3.2 Client

Der Client unterscheidet sich etwas von den restlichen Knoten, da er keine Unterklasse von BaseNode darstellt, sondern eine eigene Klasse. Mit ihm soll eine Client-Netzwerk Interaktion simuliert werden, indem Proposals, welche gestellt werden sollen zuerst von einem Client an einen Proposer gesendet werden, welcher diese dann entweder vertritt, oder ablehnt.

4 Analyse und Evaluation

4.1 Beispiel

In einem Testversuch wurde ein simples Netzwerk, bestehend aus einem Proposer, drei Acceptors, einem Learner und einem Client aufgebaut und ein Wert ”test” durch den Client angefragt. Das ergebnis der Logs wurde anschließend in ein Sequenzdiagramm in Abbildung 5 umgewandelt.

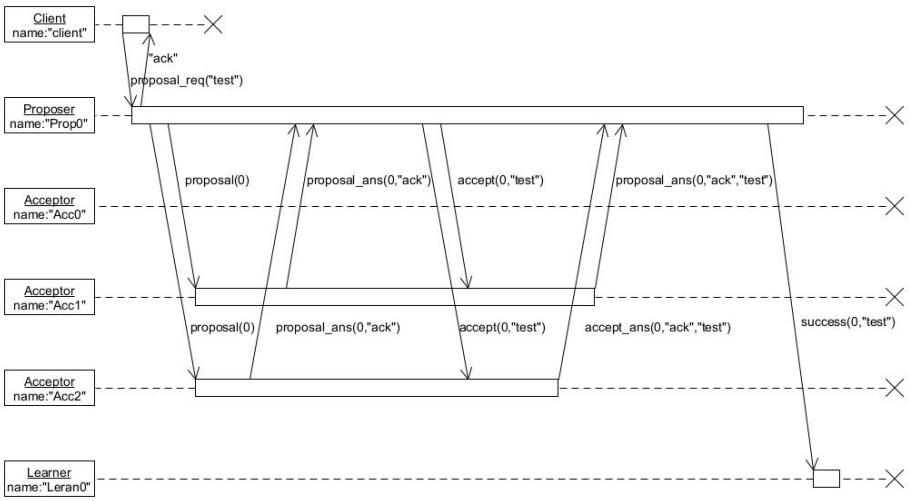

Abbildung 5: Sequenzdiagramm eines Acceptor Ausfalls

4.2 Vergleich zur Theorie

Der im Rahmen dieser Arbeit erstellten Implementierung fehlt es an manchen Stellen an Detail und Präzision. Es gibt beispielsweise keine Möglichkeit den Ausfall eines Knoten zu simulieren und Nachrichten kommen immer Garantiert an (Bit-Flips durch kosmische Strahlung o.ä. ausgenommen). Der Verwendung der Klasse PaxosNetwork umgeht zudem die Problematik, dass neue Proposer sich im Normalfall erst einmal Informationen über die aktuellen Acceptors verschaffen müssen, da dies im Normalfall auch über Nachrichten im Netzwerk geschieht. Weiterhin wurde nur der Fall diskutiert, in welchem nur ein Proposer eine Anfrage stellt. Insgesamt kann sich, über die Implementierung, die allgemeine Funktionsweise von Paxos dennoch in nachvollziehbarer Weise veranschaulicht werden.

Literatur

[1] Leslie Lamport. The part-time parliament. *ACM Trans. Comput. Syst.*, 16(2):133–169, May 1998.

[2] Leslie Lamport. Paxos made simple. 2001.