Institut für Architektur von Anwendungssystemen

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Masterarbeit

Die Rolle von Verschränkung im Quantencomputing: Speedup und Konsensusprotokolle

Marie Olivia Salm

Studiengang:	Informatik
Prüfer/in:	Prof. Dr. rer. nat. Dr. h. c. Frank Leymann
Betreuer/in:	Daniel Vietz, M.Sc.
Beginn am:	7. Mai 2019
Beendet am:	7. November 2019

Kurzfassung

In Zukunft werden Quantencomputer Probleme womöglich effizienter lösen als klassische Computer. Dies wäre eine bahnbrechende Errungenschaft und erweckt daher große Hoffnungen bei Forschung und Wirtschaft. Noch befindet sich das Gebiet der Quanteninformatik und des Quantencomputings vor allem in der Grundlagenforschung, und die Entwicklung eines leistungsfähigen Quantencomputers liegt noch in weiter Ferne. Dennoch werden bereits heute Quantenalgorithmen entwickelt, die eine Überlegenheit gegenüber klassischen Algorithmen aufzeigen könnten. So könnten verteilte Systeme von den quantenmechanischen Eigenschaften unter anderem durch Kommunikationsersparnisse profitieren. In dieser Arbeit wurde untersucht, ob das Phänomen der Verschränkung für den möglichen Speedup gegenüber klassischen Computer verantwortlich ist. Dazu wurden Annahmen wissenschaftlicher Arbeiten zusammengefasst. Des Weiteren wurde das Konsensusprotokoll Paxos mit quantenmechanischen Konzepten erweitert. Für eine der Erweiterungen wurde der verschränkte W-Zustand für die Wahl eines Proposers eingesetzt. In der zweiten Erweiterung wurde für die Bestimmung einer Rundennummer Superposition verwendet. Zudem wurde das 2-Phasen-Commit-Protokoll in unterschiedlichen Varianten mit dem GHZ-Zustand erweitert. Auch für das 3-Phasen-Commit-Protokoll wurde der W-Zustand für die Wahl eines Koordinators verwendet. Die Ergebnisse zeigen unter anderem, dass eine Reduzierung des Kommunikationsaufwands bei Paxos und dem 3-Phasen-Commit-Protkoll möglich ist. Es zeigt sich auch, dass eine Deblockierung des erweiterten 2-Phasen-Commit-Protokolls in der behandelten Weise nicht möglich ist.

Inhaltsverzeichnis

Abbildungsverzeichnis

Verzeichnis der Algorithmen

Abkürzungsverzeichnis

ACID Atomicity, Consistency, Isolation, Durability

BPP bounded error probabilistic polynomial time

BQP bounded error quantum polynomial time

CNOT controlled-Not

EPR Einstein, Podolsky, Rosen

GHZ Greenberger, Horne, Zeilinger

NISQ Noisy Intermediate-Scale Quantum

QAOA Quantum Approximate Optimization Algorithm

Qubit Quantenbit

VQE Variational Quantum Eigensolver

1 Einleitung

Die Quantenmechanik ist eine der wichtigsten Theorien der Physik [Haa15]. Sie hat ihren Ursprung im frühen 20. Jahrhundert. Nur kurze Zeit später, Ende des 20. Jahrhunderts, entstand die Verknüpfung der Quantenmechanik mit einer weiteren bahnbrechenden Theorie, der Informatik [RP11]. Daraus resultierend entstand die Quanteninformatik, welche sich unter anderem aus den Bereichen Quantenkryptographie, Quantenkommunikation und Quantencomputing zusammensetzt. Während das Bit die Grundlage für die Informatik und den klassischen Computer, wie er heutzutage in Verwendung ist, bildet, ist die Grundlage für die Quanteninformatik und den sogenannten Quantencomputer das Quantenbit [Hom18; NC11; RP11]. Dabei können Quantenbits aus physikalischer Sicht beispielsweise Atome, Elektronen oder Photonen sein [Pre18]. So unterliegen Quantencomputer dem Gesetz der Quantenmechanik, während der klassische Computer der klassischen Mechanik entspricht [RP11].

Bereits Ende des 20. Jahrhunderts zeigte sich, dass die Quantenmechanik Eigenschaften aufzeigt, die mit Hilfe eines klassischen Computers nicht effizient simuliert werden können [Fey82; RP11]. Quantenbits können auf eine Art miteinander korrelieren, wie es so in der klassischen Welt nicht möglich ist [Hom18; NC11; RP11]. Um diese Korrelation mit einem klassischen Computer beschreiben zu können, wird ein Vielfaches an Bits benötigt [Pre18]. So besteht die Hoffnung, dass durch diese und weitere Besonderheiten der Quantenmechanik Quantencomputer mächtiger sind und damit Probleme besser lösen können als klassische Computer.

Nach aktuellen Erkenntnissen kann jedoch die Hoffnung auf mehr Effizienz und Mächtigkeit durch Quantencomputer in naher Zukunft nicht erfüllt werden [Bro19; Nat19; Pre18]. Es hat sich herausgestellt, dass Quantenbits nur schwer zu kontrollieren sind und Berechnungen mit Quantencomputern stark von der Umwelt beeinflusst werden, sodass Fehler auftreten [Bro19; Haa15; RP11]. Des Weiteren ist bisher nicht bewiesen, dass Quantencomputer im Allgemeinen tatsächlich mächtiger als klassische Computer sind [NC11; RP11].

Weiterhin müssen die Grundlagen der Quantenmechnik erforscht werden, um ein größeres Verständnis zu erlangen [Haa15]. So hat sich in den letzten Jahrzehnten ein neues, grundlegendes und aufregendes Forschungsgebiet entwickelt, welches neben großem wissenschaftlichem auch wirtschaftliches Interesse geweckt hat [Haa15; Pre18].

Es hat sich gezeigt, dass korrelierende Quantenbits örtlich voneinander getrennt werden können und deren Korrelation auch über weite Entfernungen hinweg noch existieren [Hom18]. Diese Besonderheit kann unter anderem in verteilten Systemen Anwendung finden, um den Kommunikationsaufwand von Konsensusprotokollen zu reduzieren, wie bereits in einigen wissenschaftlichen Arbeiten untersucht wurde [BH05; CKS10; DP04]. Konsensusprotokolle haben die Aufgabe, die Konsistenz in einem hochverfügbaren verteilten System trotz aufkommender Fehler sicherzustellen [Lam96; VA15]. Für den Erhalt eines solchen hochverfügbaren Systems werden mehrere Server mit der gleichen Funktionalität bereitgestellt [Lam96]. Um die gleiche Funktion auszuführen und Konsistenz sicherzustellen, müssen die Server mit Hilfe eines Konsensusprotokolls stets einen

gemeinsamen Konsens finden. Abhängig davon, in welcher Umgebung sich das verteilte System befindet und welche Art von Fehlern angenommen werden, besteht eine gewisse Schwierigkeit die Forderung der Konsistenz zu erfüllen [Lam96; VA15].

Ein Ziel dieser Arbeit ist es zu untersuchen, ob die auftretenden Korrelationen zwischen Quantenbits tatsächlich die Ursache für die möglicherweise effizientere Berechnung von Quantencomputern ist. Dafür werden die Meinungen und Erkenntnisse verschiedener wissenschaftlicher Arbeiten zusammengetragen und aufgezeigt.

Ein weiteres Ziel ist es, Konsensusprotokolle für Quantencomputer zu entwickeln. Dabei soll untersucht werden, inwiefern die Eigenschaften der Quantenmechanik den Kommunikationsaufwand in Konsensusprotokollen für verteilte Systeme reduzieren können. Dazu wird das klassische Konsensusprotokoll Paxos [Lam98] mit quantenmechanischen Konzepten erweitert. Anschließend werden Vor- und Nachteile des Entwickelten gegenübergestellt. Schließlich werden gewonnenen Erkenntnisse dokumentiert.

Als eine weitere spezielle Art von Konsensusprotokollen werden das 2-Phasen-Commit- sowie das 3-Phasen-Commit-Protokoll mit quantenmechanischen Erweiterungen konzipiert und auf besondere Merkmale hin untersucht. So soll untersucht werden, ob das 2-Phasen-Commit-Protokoll mit Hilfe von Quantenmechanik deblockiert werden kann. Anhand einer quantenmechanischen Erweiterung für das 3-Phasen-Commit-Protokoll soll untersucht werden, ob sich Reduzierungen im Kommunikationsaufwand ergeben können. Auch hier werden bei beiden Commit-Protokollen die jeweils entstehenden Vor- und Nachteile gegenübergestellt und gewonnene Erkenntnisse dokumentiert.

Gliederung der Arbeit

Die Arbeit ist in folgender Weise gegliedert:

Kapitel 2 – Grundlagen und verwandte Arbeiten Behandeln der Grundlagen und Vorstellen verwandter Arbeiten. Genauer betrachtet werden die Themen Quanteninformatik, Konsensusprotokolle und Commit-Protokolle.

Kapitel 3 – Quantencomputing in der Praxis Darstellung der aktuellen Situation innerhalb des Forschungsgebiets.

Kapitel 4 – Speedup durch Verschränkung? Aufzeigen der verschiedenen Annahmen, die zu einem Speedup führen können.

Kapitel 5 – Verwendung von Quantencomputing in Konsensusprotokollen Präsentation zweier Erweiterungen für Paxos. Dabei werden die Eigenschaften des W-Zustands und der Superposition betrachtet.

Kapitel 6 – Verwendung von Quantencomputing in Commit-Protokollen Präsentation von Erweiterungen für Commit-Protokolle. Dabei werden die Eigenschaften des GHZ- und W-Zustands betrachtet.

Kapitel 7 – Zusammenfassung und Ausblick Zusammenfassen der Erkenntnisse aus den Bereichen Speedup, Konsensusprotokolle und Commit-Protokolle. Ausblick auf den Einsatz der vorgestellten Protokolle.

2 Grundlagen und verwandte Arbeiten

In diesem Kapitel wird auf die Grundlagen und verwandte Arbeiten des Themenbereichs der Arbeit eingegangen. Dabei werden zunächst die Grundlagen der Quanteninformatik behandelt. Anschließend wird das Aufgabengebiet von Konsensusprotokollen im verteilten System erläutert und auf ausgewählte Protokolle eingegangen, die sowohl für klassische, als auch für Quantencomputer entwickelt wurden. Zuletzt werden Commit-Protokolle aus dem Bereich der Datenbanken behandelt.

2.1 Quanteninformatik

In den folgenden Abschnitten werden angefangen bei Quantenbits in Abschnitt 2.1.1 bis hin zu dem Prinzip eines Quantenalgorithmus in Abschnitt 2.1.10 die Grundlagen der Quanteninformatik erläutert. Dabei werden unter anderem Quantenregister in Abschnitt 2.1.3, Operatoren in Abschnitt 2.1.5 und der Vorgang bei einer Messung in Abschnitt 2.1.8 erklärt. Des Weiteren wird auf die Besonderheit der sogenannten Verschränkung, siehe Abschnitt 2.1.6, eingegangen. Zudem werden Quantenalgorithmen mit signifikantem Speedup in Abschnitt 2.1.12 vorgestellt. Zuletzt wird die Bedeutung der Komplexitätsklasse *BQP* erläutert, zu sehen in Abschnitt 2.1.13.

2.1.1 Qubit

Das wohlbekannte Bit bildet die Grundlage für die klassische Informatik heutiger Computer [Hom18; NC11]. Dabei befindet sich das klassische Bit entweder im Zustand 0 oder 1. Ein Analogon in der Quanteninformatik ist das Quantenbit oder auch Qubit als Grundlage der Quantencomputer [NC11]. Ein Qubit kann im Quantenzustand $|0\rangle$, $|1\rangle$ oder einer Linearkombination der beiden Zustände sein [NC11; RP11]. Bei diesen Zuständen handelt es sich um Vektoren, welche in der Quantenmechanik standardmäßig mit der Klammerung der sogenannten Dirac-Notation dargestellt werden [RP11]. So entspricht ein Qubit im Zustand $|0\rangle$ oder $|1\rangle$ jeweils dem Zustand 0 oder 1 eines klassischen Bits [NC11].

Die beiden Quantenzustände $|0\rangle$ und $|1\rangle$ sind linear unabhängig und bilden eine Orthonormalbasis $\{|0\rangle, |1\rangle\}$ im Vektorraum [NC11]. Quantenzustände in dieser Standardbasis können wie folgt dargestellt werden [Hom18; NC11; RP11]:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{2.1}$$

$$= \alpha \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \beta \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \tag{2.2}$$

Bei Gleichung (2.2) handelt es sich um die Darstellung mit Vektoren. Die Koeffizienten α und β sind komplexe Zahlen und werden Amplituden genannt [Hom18]. Um die Vektorlänge eines Zustands zu normieren, gilt [NC11]:

$$|\alpha|^2 + |\beta|^2 = 1 \tag{2.3}$$

Sind beide Koeffizienten ungleich 0, ist der Zustand $|\psi\rangle$ eine Linearkombination und wird in der Standardbasis als Superposition bezeichnet [RP11]. So kann sich der Zustand eines Qubit zwischen $|0\rangle$ und $|1\rangle$ befinden, wodurch unendlich viele Zustände angenommen werden können [Hom18; NC11]. Ein Quantenzustand wird somit durch einen Einheitsvektor in einem zweidimensionalen, komplexen Vektorraum beschrieben [NC11; RP11].

2.1.2 Messung eines Qubits

Klassische Bits können gelesen werden um herauszufinden, ob sie 0 oder 1 sind [Hom18]. Um hingegen den Zustand eines Qubits genau ablesen zu können, muss es gemessen werden. Hierbei wird das Anwenden eines bestimmten Messgatters gemeint, dessen Ausgabe auf einer Anzeige ablesbar ist [Mer07]. Das Ergebnis der Messung ist dabei abhängig von den Amplituden des Zustands. Der Wert $|0\rangle$ in der Gleichung (2.1) wird mit der Wahrscheinlichkeit $|\alpha|^2$ gemessen, der Wert $|1\rangle$ mit der Wahrscheinlichkeit $|\beta|^2$ [RP11]. Wenn in der Gleichung (2.1) $\alpha = 1$ ist, sodass $\beta = 0$ ist, befindet sich das Qubit mit hundertprozentiger Wahrscheinlichkeit im Zustand $|0\rangle$ [Hom18]. Werden die Werte von α und β getauscht, ergibt sich mit gleicher Wahrscheinlichkeit der Zustand $|1\rangle$. Befindet sich ein Zustand in einer Superposition, wird diese durch die Messung zerstört, und der Zustand kollabiert entweder auf den Wert $|0\rangle$ oder $|1\rangle$, entsprechend den Zuständen eines klassischen Bits [Hom18; NC11]. Bei Messung des Zustands

$$|\psi\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle \tag{2.4}$$

in der Standardbasis $\{|0\rangle, |1\rangle\}$ ergibt sich durch die Quadrierung der Amplituden im Betrag ($|\frac{1}{\sqrt{2}}|^2 = \frac{1}{2}$) jeweils eine fünfzigprozentige Chance den Wert $|0\rangle$ oder den Wert $|1\rangle$ zu erhalten [NC11]. Wird nun jedoch die sogenannte Hadamardbasis $\{\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)\}$ verwendet, wird aus dem probabilistischen ein deterministisches Messen [RP11]. So hängt es von der verwendeten Basis ab, ob von einer Superposition gesprochen und von welchem Messverhalten ausgegangen werden kann. Durch die Veränderung des Zustands bei einer Messung kann keine Messwiederholung auf den Ursprungszustand des Qubits vorgenommen werden, der Vorgang ist irreversibel [Mer07; RP11].

2.1.3 Quantenregister

Quantenregister sind analog zu klassischen Registern Folgen von Qubits [Hom18]. Während ein klassisches Register mit n Bits einen der 2^n Zustände von 0 bis $2^n - 1$ annehmen kann, kann sich ein Quantenregister mit n Qubits in einer Superposition dieser 2^n Zustände befinden. Mit der Kombination der n Qubits, dessen Zustände jeweils durch zweidimensionale Vektoren beschrieben werden, entsteht ein 2^n-dimensionaler Vektorraum [RP11]. Damit wächst die Anzahl der Dimensionen des Vektorraums exponentiell zur Anzahl der Qubits. Für eine solche Kombination an Vektoren wird das Tensorprodukt verwendet.

Seien V und W zwei Vektorräume mit jeweils der Basis $\{v_0, ..., v_{n-1}\}$ und $\{w_0, ..., w_{m-1}\}$ [Hom18; RP11]. Dann bildet das Tensorprodukt $V \otimes W$ einen $(n \cdot m)$-dimensionalen Vektorraum mit der Basis $\{v_i \otimes w_j | 0 \leq i < n, 0 \leq j < m\}$. Gegeben seien die beiden Vektoren v aus V und w aus W [Hom18]:

$$v = \sum_{i=0}^{n-1} \alpha_i v_i = \begin{pmatrix} \alpha_0 \\ \vdots \\ \alpha_{n-1} \end{pmatrix}; \qquad w = \sum_{j=0}^{m-1} \beta_j w_j = \begin{pmatrix} \beta_0 \\ \vdots \\ \beta_{m-1} \end{pmatrix} \tag{2.5}$$

Wird nun aus den beiden Vektoren das Tensorprodukt gebildet, ergibt sich ein Vektor, dessen Dimension aus dem Produkt der beiden Dimensionen n und m hervorgeht [Hom18]:

$$v \otimes w = (\sum_{i=0}^{n-1} \alpha_i v_i) \otimes (\sum_{j=0}^{m-1} \beta_j w_j) = \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} \alpha_i \beta_j (v_i \otimes w_j) = \begin{pmatrix} \alpha_0\beta_0 \\ \vdots \\ \alpha_0\beta_{m-1} \\ \alpha_1\beta_0 \\ \vdots \\ \alpha_{n-1}\beta_{m-1} \end{pmatrix} \tag{2.6}$$

Hierbei wird jedes Element α_i mit jedem Element β_j der Vektoren v und w multipliziert. Um das Tensorprodukt aus Matrizen bilden zu können, wird jedes Element der ersten Matrix mit der vollständigen zweiten Matrix multipliziert [Hom18].

Für die Entwicklung eines Quantenregisters wird das Tensorprodukt auf die zugehörigen Qubits angewendet [Hom18]. Seien beispielsweise die beiden Qubits $|x\rangle = \beta_0|0\rangle + \beta_1|1\rangle$ und $|y\rangle = \gamma_0|0\rangle + \gamma_1|1\rangle$ in der Basis $\{|0\rangle, |1\rangle\}$ gegeben. Um den Zustand $|\phi\rangle$ des Quantenregisters darstellen zu können, wird das Tensorprodukt aus $|x\rangle$ und $|y\rangle$ gebildet [Hom18]:

$$
\begin{aligned}
|\phi\rangle &= |x\rangle \otimes |y\rangle && (2.7)\\
&= (\beta_0|0\rangle + \beta_1|1\rangle) \otimes (\gamma_0|0\rangle + \gamma_1|1\rangle) && (2.8)\\
&= \beta_0\gamma_0(|0\rangle \otimes |0\rangle) + \beta_0\gamma_1(|0\rangle \otimes |1\rangle) + \beta_1\gamma_0(|1\rangle \otimes |0\rangle) + \beta_1\gamma_1(|1\rangle \otimes |1\rangle) && (2.9)\\
&= \alpha_{00}(|0\rangle \otimes |0\rangle) + \alpha_{01}(|0\rangle \otimes |1\rangle) + \alpha_{10}(|1\rangle \otimes |0\rangle) + \alpha_{11}(|1\rangle \otimes |1\rangle) \qquad ,\alpha_{ij} = \beta_i\gamma_j && (2.10)\\
&= \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle && (2.11)\\
&= \alpha_0|0\rangle + \alpha_1|1\rangle + \alpha_2|2\rangle + \alpha_3|3\rangle && (2.12)
\end{aligned}
$$

Zu sehen ist, dass die Rechenschritte in der Dirac-Notation der herkömmlichen Produktbildung ähneln [Hom18]. Die Gleichungen (2.11) und (2.12) dienen der einfacheren Lesbarkeit, wobei es sich bei der Gleichung (2.12) um die Dezimaldarstellung handelt. Die daraus entstehende Basis des 2-Qubit-Registers ist [Hom18; RP11]:

$$
\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\} = \{\begin{pmatrix}1\\0\end{pmatrix} \otimes \begin{pmatrix}1\\0\end{pmatrix}, \begin{pmatrix}1\\0\end{pmatrix} \otimes \begin{pmatrix}0\\1\end{pmatrix}, \begin{pmatrix}0\\1\end{pmatrix} \otimes \begin{pmatrix}1\\0\end{pmatrix}, \begin{pmatrix}0\\1\end{pmatrix} \otimes \begin{pmatrix}0\\1\end{pmatrix}\} = \{\begin{pmatrix}1\\0\\0\\0\end{pmatrix}, \begin{pmatrix}0\\1\\0\\0\end{pmatrix}, \begin{pmatrix}0\\0\\1\\0\end{pmatrix}, \begin{pmatrix}0\\0\\0\\1\end{pmatrix}\} \tag{2.13}
$$

Wird das Quantenregister auf n Qubits erweitert, ergibt sich die Standardbasis $\{|0\rangle, |1\rangle, |2\rangle, ..., |2^n-1\rangle\}$ [RP11]. So ergibt sich die Formulierung des Zustands $|\phi\rangle$ des Registers mit n Qubits, wobei es sich bei i um die Dezimaldarstellung, wie in Gleichung (2.12), handelt [Hom18]:

$$
|\phi\rangle = \sum_{i=0}^{2^n-1} \alpha_i|i\rangle \tag{2.14}
$$

Dabei gilt auch bei den Zuständen von Quantenregistern für die Summe der Amplituden $\sum_{i=0}^{2^n-1} |\alpha_i|^2 = 1$ [Hom18].

2.1.4 Messung eines Quantenregisters

Bei einer Messung eines Quantenregisters wird analog zur Messung eines Qubits bei Betrachtung der Zustandsgleichung (2.14) aus Abschnitt 2.1.3, mit Wahrscheinlichkeit $|\alpha_i|^2$ der Wert $|i\rangle$ gelesen, für $0 \leq i \leq 2^n - 1$ [NC11]. Die enthaltenen Qubits können zudem einzeln gemessen werden [Hom18; NC11]. Wird beispielsweise im Quantenzustand in Gleichung (2.11) das erste Qubit $|x\rangle$ gemessen, ergibt sich mit der Wahrscheinlichkeit $|\alpha_{00}|^2 + |\alpha_{01}|^2$ der Wert $|x\rangle = |0\rangle$ [NC11]. Das Register befindet sich daraufhin in folgendem Zustand [Hom18]:

$$
|\phi'\rangle = \frac{\alpha_{00}|00\rangle + \alpha_{01}|01\rangle}{\sqrt{|\alpha_{00}|^2 + |\alpha_{01}|^2}} \tag{2.15}
$$

Hierbei dient $\sqrt{|\alpha_{00}|^2 + |\alpha_{01}|^2}$ der erneuten Normalisierung nach der Messung [NC11]. Über die Amplituden des Folgezustands wird nichts weiter bekannt [Hom18]. Wird nun im zweiten Schritt das zweite Qubit $|y\rangle$ gemessen, ist $|00\rangle$ mit Wahrscheinlichkeit $\frac{\alpha_{00}}{|\alpha_{00}|^2+|\alpha_{01}|^2}$ das Ergebnis. Dementsprechend ist die Wahrscheinlichkeit für den Wert $|11\rangle$. Analog verhält sich der Zustand des Registers mit dem Messergebnis $|x\rangle = |1\rangle$ sowie dem zweiten Qubit $|y\rangle$ als Erstmessung. Die Reihenfolge der Messungen hat keinen Einfluss auf das Ergebnis.

2.1.5 Operatoren

Um Operationen auf Qubits ausführen zu können, werden quadratische Matrizen verwendet [Hom18]. Diese Matrizen müssen unitär sein, damit auch für den Folgezustand nach einem Rechenschritt die Normierungsbedingung aus Gleichung (2.1.3) gilt [NC11]. Sei $\overline{A}$ das komplex konjungierte einer Matrix A und A^T die transponierte [Hom18]. Dabei ist $A^* = (\overline{A})^T$ die zu A adjungierte Matrix. So ist eine Matrix unitär genau dann, wenn $A^*A = I$, wobei I die Einheitsmatrix ist [NC11]. Bei einer unitären Matrix gilt zudem, dass das Inverse gleich der zu A adjungierten ist, $A^{-1} = A^*$ [Hom18]. Womit jeder Rechenschritt mit einer unitären Matrix umkehrbar ist [Hom18; RP00]. Unitäre Matrizen, auf die Zustandsvektoren von Qubits angewendet, sind unitäre Abbildungen der Vektoren oder auch unitäre Transformationen [Hom18]. Zusammengefasst sind unitäre Transformationen winkelerhaltend, längenerhaltend, linear und reversibel [Hom18; RP00].

Wichtige unitäre Zustandstransformationen für ein Qubit sind die Pauli-Matrizen [NC11]:

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}; \quad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}; \quad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{2.16}$$

X negiert das Qubit, Z ist ein Phasenflip, da die zweite Amplitude negiert wird, und für Y gilt, $Y = ZX$ [Hom18; RP00]. Führt man diese sogenannten Quantengatter auf dem Zustand $\alpha|0\rangle + \beta|1\rangle$ aus, ergibt sich [Hom18]:

$$X : \alpha|0\rangle + \beta|1\rangle \mapsto \beta|0\rangle + \alpha|1\rangle \tag{2.17}$$
$$Y : \alpha|0\rangle + \beta|1\rangle \mapsto -i\beta|0\rangle + i\alpha|1\rangle \tag{2.18}$$
$$Z : \alpha|0\rangle + \beta|1\rangle \mapsto \alpha|0\rangle - \beta|1\rangle \tag{2.19}$$

Eine weitere wichtige unitäre 1-Qubittransformation ist die Hadamard-Matrix [NC11; RP00]:

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{2.20}$$

Auf den Zustand $|0\rangle$ angewendet, erhält man eine gleichverteilte Superposition, wie in Abschnitt 2.1.2 Gleichung (2.4) [RP00]:

$$H : |0\rangle \mapsto \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \tag{2.21}$$

$$|1\rangle \mapsto \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \tag{2.22}$$

Erweitert auf ein Quantenregister mit n Qubits, wird ein Tensorprodukt aus n Hadamard-Matrizen gebildet, und eine Superposition aus allen 2^n verschiedenen Zuständen entsteht [Hom18; RP00]:

$$(H \otimes ... \otimes H)|0...0\rangle = \frac{1}{\sqrt{2^n}}((|0\rangle + |1\rangle) \otimes ... \otimes (|0\rangle + |1\rangle)) \tag{2.23}$$

$$= \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle \tag{2.24}$$

Des Weiteren werden kontrollierte Gatter verwendet, um *if*-Bedingungen zu realisieren [Hom18; NC11]. Ein Beispiel ist CNOT oder auch *controlled*-NOT [NC11]:

$$\text{CNOT} : |c,t\rangle \mapsto |c, t \oplus c\rangle \tag{2.25}$$

Hierbei ist $|c\rangle$ das *Control*-Qubit und $|t\rangle$ das *Target*-Qubit. Für den Fall, dass $|c\rangle = |1\rangle$ ist, wird der Wert von $|t\rangle$ negiert, ansonsten bleibt $|t\rangle$ unverändert [Hom18; NC11]. Bei Anwendung des CNOT-Gatters auf die Superposition eines 2-Qubit-Registers wie in Abschnitt 2.1.3 Gleichung (2.11), ergibt sich [Hom18]:

$$\alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle \xmapsto{\text{CNOT}} \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|11\rangle + \alpha_{11}|10\rangle \tag{2.26}$$

Der dritte und vierte Wert wird durch das CNOT getauscht, die Reihenfolge der Amplituden verändert sich nicht. Solch ein Gatter mit zwei Eingabequbits lässt sich mit einem beliebigen, unitären 1-Qubitoperator U verallgemeinern [NC11]. Wenn $|c\rangle = |1\rangle$ ist, wird U auf das *Target*-Qubit $|t\rangle$ angewendet, *controlled-U* : $|c\rangle|t\rangle \mapsto |c\rangle U^c|t\rangle$. Zudem ist es möglich, mit Hilfe von CNOT-Gattern und 1-Qubitoperatoren alle unitären Transformationen für n Qubits umzusetzen [RP11].

Werden auf einem Quantenregister mit $n + 1$ Qubits, die sich im Zustand $|0\rangle^{\otimes n}|0\rangle$ befinden, zunächst n Hadamard-Matrizen angewendet, ergibt sich der Zustand $\frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle|0\rangle$, siehe auch Gleichung (2.24) [NC11; RP11]. Beim anschließenden Anwenden einer Transformation $|x, y\rangle \mapsto |x, y \oplus f(x)\rangle$ mit $f(x) : \{0,1\} \mapsto \{0,1\}$ wird der Zustand $\frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle|f(x)\rangle$ erzeugt. Durch den vorherigen Zustand $|0\rangle$ des letzten Qubits ist es nach der Transformation im Zustand $|f(x)\rangle$ und abhängig davon, ob $f(x)$ 0 oder 1 ist [NC11]. $f(x)$ ist mit x verschränkt, und durch die Superposition sind alle 2^n Funktionswerte von $f(x)$ allein durch einmaliges Anwenden der Transformation gleichzeitig vorhanden [NC11; RP11]. Dieses Vorkommnis wird Quantenparallelismus genannt. Für

das Lesen der Informationen muss allerdings gemessen werden, welches wiederum die Superposition zerstört und, gemessen in der Standardbasis, nur das Ergebnis für einen zufälligen Wert von x ausgibt. So bringt Quantenparallelismus selbst keinen Vorteil [RP11]. Um Quantenparallelismus sinnvoll nutzen zu können, bedarf es weiterer Transformationen.

Eine bedeutende unitäre Transformation ist das Toffoli-Gatter [NC11]:

$$\text{Tof} : |a, b, c\rangle \mapsto |a, b, c \otimes ab\rangle \tag{2.27}$$

Dabei sind a und b die *Control*-Qubits, und c ist das *Target*-Qubit [NC11]. c wird negiert, wenn sowohl a als auch b auf 1 gesetzt ist. Die anderen Fälle haben keine Auswirkung auf c. Das Toffoli-Quantengatter ermöglicht es, laut Nielsen und Chuang, klassische NAND-Gatter zu simulieren [NC11]. Dafür wird c auf 1 gesetzt, und es ergibt sich $|a, b, 1 \otimes ab\rangle = \neg(a \wedge b)$. Es ist im Allgemeinen bekannt, dass mit NAND-Gattern alle anderen klassischen Gatter erzeugt werden können. Allerdings ist beispielsweise das NAND-Gatter nicht umkehrbar [NC11]. Mit Hilfe der Ausgabe des Gatters lässt sich nicht eindeutig die Eingabe rekonstruieren [Hom18]. Die zwei Eingabebits mit den Werten (0,0), (0,1) und (1,0) ergeben stets die Ausgabe 1. So wird aus solch einer unumkehrbaren Schaltung durch Einsatz des Toffoli-Gatters eine umkehrbare [NC11]. Dieses Quantengatter beweist, dass Quantencomputer alle Berechnungen klassischer Computer durchführen können.

Zu beachten ist, dass es nicht möglich ist, einen unbekannten Quantenzustand zuverlässig mit Hilfe von unitären Transformationen zu kopieren [NC11; RP11]. Die Rede ist vom sogenannten No-Cloning-Theorem [WZ82]. So können zwei unterschiedliche Quantenzustände nur kopiert werden, wenn sie orthogonal zueinander sind [Hom18; NC11]. Klassische Zustände können hingegen immerzu kopiert werden, da sie unabhängig von der Basis orthogonal zueinander sind [Hom18].

2.1.6 Verschränkung

Sei ein 2-Qubit-Register im Zustand $|00\rangle$ gegeben [Hom18]. Zunächst wird die Hadamard-Matrix auf das erste Qubit angewendet, indem das Tensorprodukt aus Hadamard-Matrix und Einheitsmatrix für das zweite Qubit auf dem Quantenregister ausgeführt wird. Darauf folgend wird CNOT auf das Register angewendet [Hom18]:

$$|00\rangle \xmapsto{H \otimes I_2} \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)|0\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |10\rangle) \xmapsto{\text{CNOT}} \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \tag{2.28}$$

Bei der Messung eines der beiden Qubits hat dieses mit jeweils fünfzigprozentiger Wahrscheinlichkeit den Wert $|0\rangle$ oder $|1\rangle$ [Hom18]. Mit $|0\rangle$ als Ergebnis geht das Quantenregister in den Zustand $|00\rangle$ über. So wird bei der Messung des nächsten Qubits mit Wahrscheinlichkeit 1 ebenfalls der Wert $|0\rangle$ gemessen. Dabei funktioniert es mit $|1\rangle$ als erstes Messergebnis analog. Die Messung des einen Qubits hat in diesem Fall Einfluss auf die Messung des anderen Qubits [RP00]. Die beiden Qubits sind miteinander *verschränkt*. Dabei ist es nicht möglich, einen verschränkten Zustand als Tensorprodukt der einzelnen Qubits zu schreiben [RP11]. Verschränkung ist zudem losgelöst von der Basiswahl. Der Zustand $\frac{1}{\sqrt{2}}(|00\rangle + |01\rangle)$ ist beispielsweise nicht verschränkt

[RP00]. Zum einen beeinflusst das Messen des einen Qubits nicht das Ergebnis des anderen, indem sich für den Folgezustand die Wahrscheinlichkeiten ändern. Zum anderen lässt sich der Zustand als Tensorprodukt der einzelnen Qubits schreiben [RP00]: $\frac{1}{\sqrt{2}}(|00\rangle + |01\rangle) = |0\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$.

Es ist möglich, die zwei verschränkten Qubits aus Gleichung (2.28) räumlich voneinander zu trennen ohne, dass sich der Zustand ändert [Hom18]. Beispielsweise erhalten die Personen Alice und Bob jeweils das erste und das zweite Qubit. Beide können nur Messungen an ihrem eigenen Qubit durchführen und dürfen keine Absprache miteinander halten [Hom18; RP11]. Für jeden ist die Wahrscheinlichkeit, den Wert $|0\rangle$ oder $|1\rangle$ zu messen, bei 50% [Hom18]. Misst einer der beiden den Wert $|0\rangle$, wird der andere ebenfalls $|0\rangle$ messen, sodass sich der Zustand $|00\rangle$ ergibt, unabhängig davon, wer zu welcher Zeit misst [Hom18; RP11]. Dasselbe gilt für den Wert $|1\rangle$. Die Qubits haben trotz der räumlichen Trennung Einfluss aufeinander [Hom18]. Erst durch eine anschließende Absprache erkennen Alice und Bob, dass sie dasselbe Messergebnis erhalten haben. Dieses Phänomen wird EPR-Paradoxon [EPR35] genannt, benannt nach den Forschern Einstein, Podolsky und Rosen. Der verwendete Zustand $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ wird auch EPR-Paar genannt. Der verschränkte Zustand $\frac{1}{\sqrt{2}}(|01\rangle + |10\rangle)$ ist ein weiteres EPR-Paar, bei dem Alice und Bob jeweils entgegengesetzte Ergebnisse erhalten.

Mit einem EPR-Paar ist es jedoch nicht möglich, schneller als Licht kommunizieren zu können, da das Messergebnis weiterhin ein Zufallsexperiment ist und im Vorhinein nicht festgelegt werden kann, ob das Ergebnis $|00\rangle$ oder $|11\rangle$ ist [RP11].

Die beiden EPR-Paare sind zwei der vier sogenannten Bell-Zustände, benannt nach dem gleichnamigen Physiker [Hom18]:

$$|\phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \tag{2.29}$$

$$|\phi^-\rangle = \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle) \tag{2.30}$$

$$|\psi^+\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle) \tag{2.31}$$

$$|\psi^+\rangle = \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle) \tag{2.32}$$

Die zwei Qubits der vier Bell-Zustände korrelieren jeweils in gleicher Weise miteinander [Hom18]. Die Bell-Zustände bilden für den Vektorraum der beiden Qubits die Orthonormalbasis. Des Weiteren sind die vier Bell-Zustände *maximal verschränkt* [Hom18; RP11]. Werden die zwei Qubits einzeln betrachtet, ist ihr Messergebnis so unsicher wie möglich. Haben die Amplituden nicht denselben Wert, sondern sei beispielsweise der Zustand $\frac{1}{2}(|00\rangle + \frac{\sqrt{3}}{2}|11\rangle)$ gegeben, wird mit Wahrscheinlichkeit 1/4 der Wert $|00\rangle$ und mit Wahrscheinlichkeit 3/4 der Wert $|11\rangle$ gemessen [Hom18]. Das Messergebnis ist hier nicht so unsicher wie möglich und somit weniger verschränkt, da der Wert $|11\rangle$ wahrscheinlicher ist.

Bei der Betrachtung von verschränkten Zuständen mit n Qubits gibt es unter anderem den GHZ-Zustand [GHZ07] und den W-Zustand [RP11]:

$$|GHZ_n\rangle = \frac{1}{\sqrt{2}}(|0...0\rangle + |1...1\rangle) \tag{2.33}$$

$$|W_n\rangle = \frac{1}{\sqrt{n}}(|0...001\rangle + |0...010\rangle + ... + |1...000\rangle) \tag{2.34}$$

Bei der Messung des GHZ-Zustands in der Standardbasis erhalten alle n Qubits mit fünfzigprozentiger Wahrscheinlichkeit entweder den Wert $|0\rangle$ oder alle den Wert $|1\rangle$. Wird der W-Zustand in der Standardbasis gemessen, ergibt sich mit jeweils der Wahrscheinlichkeit $\frac{1}{n}$ einer der n Zustände, bei dem genau ein Qubit den Wert $|1\rangle$ erhält und alle anderen den Wert $|0\rangle$. Um diese beiden Zustände genauer betrachten und miteinander bezüglich der Verschränkung vergleichen zu können, wird die *Persistenz* sowie die *Verbundenheit* verwendet [RP11]. Die Persistenz eines verschränkten Systems beschreibt die kleinstmögliche Anzahl an Qubits, die gemessen werden muss, um mit vollkommener Sicherheit von einem unverschränkten Zustand ausgehen zu können. Wird im GHZ-Zustand bereits ein Qubit beispielsweise mit dem Ergebnis $|0\rangle$ gemessen, befindet sich das Quantenregister im unverschränkten Zustand $|0...0\rangle = |0\rangle \otimes ... \otimes |0\rangle$, die Persistenz ist 1. Um im W-Zustand die Verschränkung garantiert zu lösen, müssen $n - 1$ Qubits gemessen werden. Dies deckt auch den Fall ab, dass erst das in der Messreihenfolge letzte Qubit den Wert $|1\rangle$ erhält.

Damit ein Zustand als maximal verbunden gilt, muss jedes beliebige Paar an Qubits, nach einer Folge von 1-Qubitmessungen der übrigen Qubits, maximal verschränkt sein [RP11]. Fallen im GHZ-Zustand bis auf zwei Qubits alle anderen weg, ergibt sich der Bell-Zustand $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$. So ist der GHZ-Zustand maximal verbunden [RP11]. Fallen hingegen im W-Zustand bis auf zwei alle Qubits weg, ergibt sich $\frac{1}{\sqrt{2}}(|00\rangle + |01\rangle + |10\rangle)$, welcher kein gültiger Zustand ist. Der W-Zustand ist somit nicht maximal verbunden [RP11]. Durch die komplementären Eigenschaften lässt sich nicht darauf schließen, welcher der beiden Zustände mehr verschränkt ist.

Um ein beliebiges Quantenregister mit n Qubits darstellen zu können, werden, wie in Gleichung (2.14) erläutert, $2^n - 1$ Amplituden benötigt [NC11; RP11]. Nach Rieffel und Polak kann ein Qubit durch eine komplexe Zahl dargestellt werden. So kann das Tensorprodukt eines unverschränkten Quantenregisters mit n Qubits durch n komplexe Zahlen dargestellt werden [RP11]. Mit $2^n \gg n$ folgt, dass die meisten Quantenzustände nicht als Tensorprodukt darstellbar und verschränkt sind. Verschränkung als solche tritt nicht in der klassischen Welt auf und ist eine einzigartige und ausschlaggebende Eigenschaft der Quantenmechanik [Hom18; NC11; RP00].

2.1.7 Quantenkanäle

Während mit klassischen Kanälen nur klassische Bits übertragen werden können, können mit Quantenkanälen zusätzlich Qubits von einem Ort zum anderen übertragen werden [GIN18; Hom18]. Die über einen Quantenkanal überlieferten Informationen befinden sich stets in Quantenzuständen, sodass klassische Bits für einen Transport zunächst in Quantenzustände umgewandelt werden müssen [GIN18]. Dafür können unter anderem orthogonale Zustände, wie beispielsweise $|0\rangle$ für 0 und $|1\rangle$ für 1, verwendet werden. Um nach der Überlieferung erneut die klassischen Informationen zu erhalten, müssen die Quantenzustände gemessen werden.

So werden nach Gyongyosi et al. Quantenkanäle für die Übermittlung klassischer Informationen genutzt, um beispielsweise vor Lauschangriffen geschützt zu sein. Hierbei werden die klassischen Informationen in nicht-orthogonale Zustände umgewandelt. Diese sind nach dem No-Cloning-Theorem, siehe Abschnitt 2.1.5, nicht fehlerfrei kopierbar.

In der Realität ist ein Quantenkanal Störungen aus der Umwelt ausgesetzt [Hom18]. Dabei werden die Amplituden des Zustands des übertragenen Qubits durch diese Wechselwirkung verändert. Der Quantenkanal ist verrauscht. Um dem entgegenzuwirken, wird der zu übertragenden Information Redundanz angefügt, sodass sich nach dem Transport diese Information wiederherstellen lässt [NC11].

2.1.8 Vorgang einer Messung

Bei Messungen von Quantensystemen werden Messoperatoren auf die Vektorräume der Systeme angewendet [NC11]. Bei der projektiven Messung werden diese Operatoren *Observablen* genannt [NC11; RP11]. Observablen sind hermitesche Matrizen [Yan07]. Eine quadratische und komplexe Matrix A ist genau dann hermitesch, wenn die zu A adjungierte $A^* = A$ ist. Oder anders ausgedrückt, wenn $A^T = \overline{A}$ gilt. Hierbei sind die Eigenwerte einer hermiteschen Matrix reell.

Sei O eine Observable, dann kann diese wie folgt dargestellt werden [NC11]:

$$O = \sum_m mP_m \tag{2.35}$$

P_m ist ein sogenannter Projektor, der auf den Eigenraum von O projiziert [NC11]. m sind die zugehörigen Eigenwerte und die möglichen Ergebnisse der Messung. Sei ein beliebiger Zustand $|\psi\rangle$ gegeben, dann ist nach Nielsen und Chuang

$$p(m) = \langle\psi|P_m|\psi\rangle \tag{2.36}$$

die Wahrscheinlichkeit, m zu messen [NC11]. Bei $\langle\psi|$ handelt es sich um den transponierten Vektor von $|\psi\rangle$ [RP11]. So bildet beispielsweise $\langle\psi||\psi\rangle = \langle\psi|\psi\rangle$ das Skalarprodukt von $|\psi\rangle$. Die Messung erzeugt den Folgezustand $|\psi'\rangle$ [NC11]:

$$|\psi'\rangle = \frac{P_m|\psi\rangle}{\sqrt{p(m)}} \tag{2.37}$$

Auf diesen Zustand können schließlich weitere Rechenschritte angewendet werden [Hom18]. Beispielhaft kann die Pauli-Matrix Z aus Abschnitt 2.1.5 Gleichung (2.16) als Observable mit den Eigenwerten +1, -1 und den Eigenvektoren $|0\rangle, |1\rangle$ verwendet werden [NC11]. Sei nun der zu messende Zustand $|\psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ gegeben. Der Wert +1 mit dem Projektor $P_1 = |0\rangle\langle 0|$ wird mit Wahrscheinlichkeit $\frac{1}{2}$ gemessen [NC11; RP11]:

$$p(1) = \langle\psi|0\rangle\langle 0|\psi\rangle \tag{2.38}$$
$$= [\frac{1}{\sqrt{2}}(\langle 0| + \langle 1|) \cdot |0\rangle] \cdot [\langle 0| \cdot \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)] \tag{2.39}$$
$$= [\frac{1}{\sqrt{2}}((1 \;\; 0) + (0 \;\; 1)) \cdot \begin{pmatrix}1\\0\end{pmatrix}] \cdot [(1 \;\; 0) \cdot \frac{1}{\sqrt{2}}(\begin{pmatrix}1\\0\end{pmatrix} + \begin{pmatrix}0\\1\end{pmatrix})] \tag{2.40}$$
$$= \frac{1}{\sqrt{2}} \cdot \frac{1}{\sqrt{2}} \tag{2.41}$$
$$= \frac{1}{2} \tag{2.42}$$

Analog dazu wird mit dem Projektor $P_{-1} = |1\rangle\langle 1|$ der Eigenwert -1 ebenfalls mit Wahrscheinlichkeit $\frac{1}{2}$ gemessen [NC11]. Wird die Messung bezüglich der Observablen Z erneut auf dem Folgezustand $|\psi'\rangle$ ausgeführt, ändert sich das Ergebnis nicht. Bei der Messung eines unbekannten Quantenzustands bleibt auch der Folgezustand unbekannt [Hom18].

2.1.9 Gemischte Zustände

Wenn der Zustand eines Quantensystems nicht bekannt ist, ist die Rede von einem gemischten Zustand [BBKM04]. Hierbei besteht ein gemischter Zustand aus mehreren reinen Zuständen [RP11]. Ein Quantensystem ist in einem reinen Zustand, wenn dieser genau bekannt ist.

Ein gemischter Zustand wird mittels Dichteoperator beschrieben [Hom18; RP11]. Während der Dichteoperator eines reinen Zustands $\rho = |\psi\rangle\langle\psi|$ ist, wird ein gemischter Zustand im Allgemeinen mit dem Dichteoperator

$$\rho = \sum_{i=1}^{n} p_i |\psi_i\rangle\langle\psi_i| \tag{2.43}$$

beschrieben [RP11]. Dabei beschreibt $p_i \geq 0$ mit $\sum_i p_i = 1$ die Wahrscheinlichkeit, die sich der gemischte Zustand im reinen Zustand $|\psi_i\rangle$ befindet [BBKM04; RP11]. Die Darstellung des Dichteoperators ist nicht einzigartig und kann sich auch aus der Summe anderer, reiner Zustände ergeben [BBKM04].

Gemischte Zustände unterscheiden sich, in Bezug auf Verschränkung, von reinen [BBKM04]. So gibt es nicht verschränkte gemischte Zustände, welche sich nicht als Tensorprodukt beschreiben lassen. Ein gemischter Zustand der beiden Quantensysteme A und B ist nicht verschränkt, wenn er sich mit

$$\rho = \sum_{i=1}^{n} p_i |\psi_i^{(A)}\rangle\langle\psi_i^{(A)}| \otimes |\psi_i^{(B)}\rangle\langle\psi_i^{(B)}| = \sum_{i=1}^{n} p_i \rho_i^{(A)} \otimes \rho_i^{(B)} \tag{2.44}$$

beschreiben lässt [BBKM04; RP11]. Es handelt sich um ein Gemisch aus nicht verschränkten Zuständen $|\psi_i^{(A)}\rangle\langle\psi_i^{(A)}| \otimes |\psi_i^{(B)}\rangle\langle\psi_i^{(B)}|$, hervorgehend aus den Systemen A und B, denen jeweils eine bestimmte Wahrscheinlichkeit p_i zugeordnet ist [RP11]. Andernfalls ist der gemischte Zustand verschränkt.

2.1.10 Prinzip eines Quantenalgorithmus

Ein Quantenalgorithmus durchläuft nach dem Standardmodell drei grundlegende Schritte [CEP+18; RP11]. Zunächst wird die zu verarbeitende Information durch einen Zustandsvektor in der Standardbasis dargestellt [CEP+18; Hom18]. Dabei kann es sich um klassische oder auch um Quanteninformationen handeln [CEP+18]. Darauffolgend werden Rechenschritte auf dem Quantensystem mit CNOT-Gattern und 1-Qubitoperatoren realisiert [RP11]. Schließlich werden ein oder mehrere Qubits des erhaltenen Zustands in der Standardbasis gemessen [NC11]. Ein Quantencomputer kann neben Quantenberechnungen auch klassische Berechnungen durchführen, wie in Abschnitt 2.1.5 gezeigt [NC11]. Jedoch ist es häufig vorteilhafter und angenehmer, klassische Berechnungen auf einem klassischen Computer durchzuführen.

Ein weiteres Konzept sind hybride Algorithmen, welche das klassische Computing und das Quantencomputing miteinander vereint [MDLH18; Pre18]. So können die aktuell störanfälligen und mit wenigen Qubits ausgestatteten Quantencomputer durch klassische Computer unterstützt werden [SAC+18]. Dadurch besteht die Möglichkeit, dass bereits in naher Zukunft große Instanzen von Optimierungsproblemen mit der momentanen Quantentechnologie zuverlässig lösbar werden [Pre18; SAC+18]. Hierbei erzeugt zunächst ein Quantencomputer einen Quantenzustand mit der gewünschten Anzahl an Qubits [Pre18]. Anschließend misst er diese und übergibt das Messergebnis einem klassischen Computer. Der klassische Computer verarbeitet das Ergebnis und übergibt daraufhin dem Quantencomputer Befehle, um Änderungen daran vorzunehmen, wie der Quantenzustand erzeugt werden soll. Das gesamte Vorgehen wird mehrmals hintereinander ausgeführt. Wird schließlich ein Quantenzustand erzeugt, der grob die Lösung enthält, terminiert es. Auf ein klassisches Problem angewendet, wird von einem Quantum Approximate Optimization Algorithm, kurz QAOA [FGG14], gesprochen. Wiederum auf ein Quantenproblem angewendet, wird von einem Variational Quantum Eigensolver, kurz VQE [MRBA16; PMS+14], gesprochen.

2.1.11 Dekohärenz

In der Realität kann ein Quantensystem nicht von der Umgebung abgeschottet werden, wie bereits in Abschnitt 2.1.7 angesprochen [Hom18]. Durch die Wechselwirkung werden Quantenzustände verändert, sodass Informationen, wie bei einer Messung, verloren gehen [CY97; Hom18]. So kann eine Superposition, aufgrund dieser sogenannten Dekohärenz, nur schwer aufrechterhalten werden und gilt daher als instabil [CY97; Hom18; Zur06]. Zeiträume, in denen Quantenzustände stabil sind, sind sehr klein [Hom18]. In diesen Zeiträumen, auch Dekohärenzzeit genannt, müssen gewünschte Quantengatter sowie Messungen angewendet werden, sodass unitäre Entwicklungen der Quantenzustände möglich sind.

2.1.12 Quantenalgorithmen mit signifikantem Speedup

Zwei der bekanntesten Quantenalgorithmen, die einen erheblichen Speedup gegenüber bisher bekannten klassischen Algorithmen aufweisen, stammen von Shor [Sho94] und Grover [Gro96] [Mon16]. Hierbei ist mit Speedup das, im Vergleich, schnellere Lösen eines Problems gemeint. So wird die Laufzeit eines Algorithmus im Allgemeinen durch die Anzahl an grundlegenden Rechenschritten, die enthalten sind, beschrieben. Bei Quantensystemen handelt es sich hierbei um die, in Abschnitt 2.1.5 angesprochenen, Quantengatter. Der Algorithmus von Shor [Sho94] entspricht einer polynomiellen Laufzeit, während alle bisher bekannten klassischen Algorithmen für das selbe Problem eine exponentielle Laufzeit benötigen [Mon16; RP11]. Dabei handelt es sich um das Faktorisierungsproblem, welches einen Integer $n = p \times q$ als Eingabe hat [Mon16]. p und q sind Primzahlen, die es zu bestimmen gilt. Das wohlbekannte Kryptosystem RSA [RSA78] baut auf der bisher angenommenen Schwierigkeit der Primfaktorzerlegung auf [RP00]. So können in Zukunft ausreichend ausgestattete Quantencomputer das Kryptosystem erfolgreich mit dem Algorithmus von Shor angreifen [Mon16; Pre18]. Der Algorithmus besteht aus einem klassischen und einem Quantenanteil [RP00]. Der klassische Anteil wird ebenso bereits in anderen Faktorisierungsalgorithmen verwendet. Hierbei wird das Faktorisierungsproblem auf das Findungsproblem der Periode einer Funktion reduziert. Anschließend wird im Quantenanteil der in Abschnitt 2.1.5 besprochene Quantenparallelismus genutzt, indem alle Funktionswerte in eine Superposition gebracht werden [RP11]. Daraufhin wird die sogenannte Quanten-Fouriertransformation angewendet, die ausschlaggebend für den Speedup gegenüber bisherigen klassischen Algorithmen ist [Hom18; RP11]. Diese präpariert den Zustand dementsprechend, sodass, nach Messung, mit hoher Wahrscheinlichkeit die Periode der Funktion ausgelesen werden kann [RP00; RP11]. Die Periode wird schließlich für die Faktorisierung von n verwendet. Die Laufzeit des Algorithmus beträgt $O((\log n)^3)$ [Mon16].

Der Algorithmus von Grover [Gro96] ermöglicht die Suche innerhalb einer unsortierten Datenbank mit N Elementen und weist dabei einen quadratischem Speedup auf [Hom18; RP11]. Während ein klassischer Algorithmus eine Laufzeit von $O(N)$ hat, da er im schlechtesten Fall alle Elemente betrachten muss, hat der Algorithmus von Grover eine Laufzeit von $O(\sqrt{N})$ [Hom18]. Er ist somit beweisbar schneller als je ein klassischer Algorithmus für dieses Problem sein kann [RP11]. Allerdings weist er auch die Schranken des Quantencomputings auf, da dessen Laufzeit bereits optimal ist. Die Datenbank wird durch eine Funktion beschrieben, die genau dann 1 ist, wenn das betrachtete Element das gesuchte ist [Hom18]. Zu Beginn des Algorithmus wird eine Superposition über alle Elemente der Datenbank erzeugt. Anschließend wird die sogenannte Amplitudenverstärkung wiederholt ausgeführt. So nutzt auch der Algorithmus von Grover Quantenparallelismus [RP11]. Bei der Amplitudenverstärkung wird die Amplitude des gesuchten Elements größer, während die Amplituden der anderen Elemente kleiner werden [Hom18]. Dabei wird zunächst die Amplitude des gesuchten Elements negiert, da genau dieses durch die oben beschriebene Funktion den Wert 1 erhält und während des Ablaufs des Algorithmus ein negatives Vorzeichen erlangt. Anschließend werden alle Amplituden am Mittelwert gespiegelt. Schließlich wird gemessen und das Ergebnis ausgelesen. Der Algorithmus ist auch auf andere, ähnliche Probleme anwendbar, beispielsweise der Suche nach dem Minimum in einem Feld [DH96] oder dem bekannten Traveling Salesman Problem. Letzteres ist ein Problem aus der Komplexitätsklasse *NP* und wird daher als nicht effizient lösbar angenommen. So können weitere Probleme in *NP* einen quadratischen Speedup durch Grovers Algorithmus erfahren. Allerdings hat ein quadratischer Speedup keine merklichen Auswirkungen auf die Laufzeit eines klassischen, exponentiellen Algorithmus [RP11]. Denn auch bei Anwenden des Algorithmus von Grover bleibt die Laufzeit exponentiell.

2.1.13 Bedeutung von BQP

Der Algorithmus von Shor [Sho94] befindet sich in der Komplexitätsklasse *BQP* [Aar10]. *BQP* steht für *bounded error quantum polynomial time* [Hom18]. So können Probleme in dieser Klasse mit einer Fehlerwahrscheinlichkeit kleiner gleich $\frac{1}{3}$ in polynomieller Zeit von Quantencomputern gelöst werden [Hom18; NC11]. Analog dazu ist die herkömmliche Klasse *BPP*, für *bounded error probabilistic polynomial time*, mit $P \subseteq BPP$ [Hom18; NC11]. Die Komplexitätsklasse *P* enthält Probleme, die in effizienter Zeit von klassischen Computern und bewiesenermaßen von Quantencomputern gelöst werden können. So gilt $P \subseteq BQP$ sowie $BPP \subseteq BQP$ [Hom18]. Wobei ungeklärt ist, ob $BPP \neq BQP$ gilt. *NP* enthält die Probleme, die sich in polynomieller Zeit überprüfen lassen. Bis heute ist nicht geklärt, ob $P = NP$ gilt [NC11]. Allerdings steht fest, dass $P \subseteq NP$ ist, da effizientes Lösen ebenso effizientes Überprüfen beinhaltet. Des Weiteren gib es die Klasse *PSPACE*, die alle Probleme enthält, die innerhalb eines polynomiellen Speicherplatzes lösbar sind [Hom18]. *PSPACE* enthält die Klasse *P*, da ein polynomieller Algorithmus nicht mehr als polynomiellen Platz benötigt. Des Weiteren gilt nach Homeister $NP \subseteq PSPACE$, da der Zeitaufwand nicht effizient ist, der benötigte Platz allerdings polynomiell bleibt [Hom18]. Zudem gilt, dass Quantencomputer keine Probleme außerhalb von *PSPACE* in polynomieller Zeit lösen können, wodurch $BQP \subseteq PSPACE$ gilt [NC11]. Zusammenfassend ergibt sich $BPP \subseteq BQP \subseteq PSPACE$. Angenommen, es würde gezeigt werden, dass $BPP \neq BQP$, Quantencomputer somit leistungsstärker wären als klassische Computer, hätte dies gleichzeitig mit dem daraus folgenden $BPP \neq PSPACE$, beziehungsweise $P \neq PSPACE$, großen Einfluss auf die klassische Komplexitätstheorie. Laut Homeister ist zudem offen, ob entweder $NP \subseteq BQP$ oder $NP \neq BQP$ gilt [Hom18]. So können Quantencomputer entweder alle oder keine Probleme in *NP* effizient lösen.

2.2 Konsensusprotokolle

Um die Verfügbarkeit eines verteilten Systems nicht durch den Ausfall eines einzelnen zentralen Servers zu gefährden, ist das Einbringen von Redundanz notwendig [Lam01; Lam96]. Eine Möglichkeit, um Redundanz in solch ein System zu bringen, ist die Replikation des vorhandenen Servers [Lam96]. So wird die Funktionalität des Servers auf mehrere, voneinander unabhängige Server kopiert [Lam01]. Statt von einem Server kann im Allgemeinen auch von einem Prozess gesprochen werden. Erhalten alle Server mit derselben Funktionalität nun dieselbe Eingabe, führen sie alle dieselben Rechenschritte durch und erzeugen dieselbe Ausgabe. Von außen betrachtet, verhält sich die Gruppe weiterhin wie ein einzelner Server [ST17]. Kommt es nun zu vereinzelten Ausfällen, ist das verteilte System weiterhin verfügbar [Lam96]. Es besteht die Möglichkeit, dass zeitgleich mehrere Clients unterschiedliche Anfragen an die Server schicken können [ST17; VA15]. Daher muss sichergestellt werden, dass alle funktionierenden Repliken die Anfragen in derselben Reihenfolge abarbeiten, um so die Konsistenz im verteilten System sicherstellen zu können [Lam01; ST17]. Um diese Bedingung zu erfüllen, müssen sich die Server auf eine Anfrage und deren enthaltene Eingabe einigen [ST17]. Für diesen Einigungsprozess wird ein sogenanntes Konsensusprotokoll herangezogen [Lam01]. Ein Konsensusprotokoll muss die drei Eigenschaften Sicherheit, Terminierung und Gültigkeit erfüllen [Asp02; Lam96]. Zum einen müssen die Server, die bei der Entscheidung mitgewirkt haben, sich für dieselbe Eingabe entscheiden [Asp02]. Zum anderen muss von allen funktionierenden Servern letztendlich eine Entscheidung getroffen werden.

Im Folgenden wird das Konsensusprotokoll Paxos [Lam98] vorgestellt, siehe Abschnitt 2.2.1. Dabei wird der Ablauf des Protokolls sowie dessen Eigenschaften präsentiert. Schließlich wird die Erweiterung Multi-Paxos in Abschnitt 2.2.2 aufgezeigt. Auf bereits bestehende Konsensusprotokolle für Quantencomputer wird anschließend eingegangen, zu sehen in Abschnitt 2.2.3.

2.2.1 Paxos

Ein bekanntes Konsensusprotokoll für klassische Computer ist Leslie Lamports Paxos [Lam98]. Hierbei schlagen Server den anderen Servern eine Eingabe vor [DLL00]. Eine Entscheidung wird getroffen, indem die Mehrheit der teilnehmenden Server für dieselbe Eingabe stimmen. Es wird davon ausgegangen, dass sich das verteilte System in einer asynchronen Umgebung befindet [VA15]. Was in diesem Kontext bedeutet, dass keine zeitlichen Bedingungen aufgestellt werden können, da alle Aktionen in einem beliebigen Zeitraum, in einer beliebigen Geschwindigkeit stattfinden [ST17; VA15]. Zudem können Nachrichten zwischen den Servern verloren gehen oder vervielfacht werden [Lam01]. So kann nicht festgestellt werden, ob ein Server ausgefallen oder nur sehr langsam ist [FLP85; VA15]. Fällt ein Server aus, hält dieser an [Lam01]. Es treten keine byzantinischen Fehler auf, bei denen Server zufällig oder bösartig handeln [Lam01; LSP82]. Entschiedene Eingaben werden persistent gespeichert, sodass nach einem Ausfall eine Wiederherstellung des letzten Zustands möglich ist [CGR07]. Des Weiteren ist die Anzahl der teilnehmenden Servern fest und bekannt [Lam01].

Grundlegend hat jeder teilnehmende Server die Möglichkeit, den anderen Servern eine Eingabe vorzuschlagen [Lam01]. Mit dem Vorschlagen einer Eingabe beginnt eine Runde [DLL00]. Der Server, der den Vorschlag gebracht hat, wird Proposer der Runde genannt [DLL00; Lam01]. An jede Runde ist eine einzigartige Rundennummer gebunden [DLL00]. Es ist möglich, dass mehrere Runden sowie Proposer zur selben Zeit existieren.

Ablauf

Zu Beginn einer Runde sendet ein Proposer eine *Prepare*-Nachricht mit der Rundennummer r, wie im Kommunikationsverlauf in Abbildung 2.1a zu sehen, an die anderen Server [DLL00; Lam01]. So holt sich dieser Informationen zu bisher laufenden Runden ein [DLL00]. Die Empfänger befinden sich nun in der Rolle des Acceptors [Lam01]. Sie antworten dem Proposer mit einer *Promise*-Nachricht, wenn r größer ist als die Rundennummern der bisher empfangenen *Prepare*-Nachrichten [CGR07; Lam01]. Hiermit zeigt der Acceptor dem Proposer der Runde, dass er keine Runden mehr kleiner r annimmt [Lam01]. Die *Promise*-Nachricht enthält die aktuelle Rundennummer r, die Rundennummer m der letzten Runde, bei der eine Eingabe akzeptiert wurde, sowie die zugehörige Eingabe v, wie in Abbildung 2.1a dargestellt [DLL00]. Wurde bisher in keiner Runde, bei der der Acceptor involviert war, eine Eingabe von ihm akzeptiert, gibt er als Rundennummer 0 und als Eingabe `nil` zurück.

Antwortet dem Proposer die Mehrheit an Acceptors, ist sicher, dass er momentan die aktuellste Runde führt und somit eine Eingabe vorschlagen kann [DLL00; Lam01]. Er versendet, wie in Abbildung 2.1a gezeigt, eine *Accept*-Nachricht an eine beliebige Mehrheit an Acceptors. Mit ihr werden die Rundennummer r und die vorzuschlagende Eingabe v übermittelt [Lam01]. Die Eingabe wird aus der empfangenen *Promise*-Nachricht mit der bisher höchsten überbrachten Rundennummer

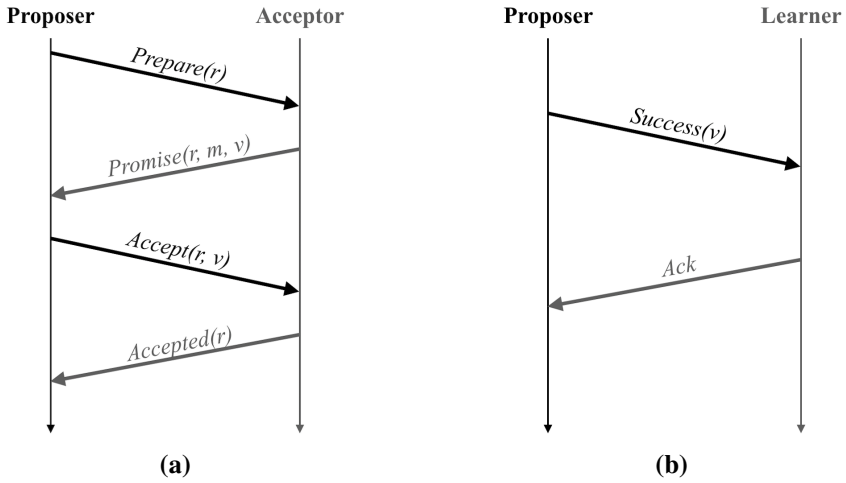

Abbildung 2.1: Kommunikationsverlauf von Paxos, orientiert an [DLL00]. (a) Nachrichtenaustausch zwischen Proposer und Acceptor zur Entscheidung einer Eingabe. (b) Bekanntgabe der entschiedenen Eingabe an Learner sowie dessen Bestätigung.

entnommen. Sind keine Eingabevorschläge vorhanden, hat der Proposer die freie Wahl [CGR07]. Auch hier prüfen die empfangenden Acceptors erneut, ob sie nicht in der Zwischenzeit eine *Prepare*-Nachricht mit Rundennummer größer r empfangen haben [Lam01]. Ist die Rundennummer r weiterhin aktuell, senden sie eine *Accepted*-Nachricht mit r an den Proposer, wie in Abbildung 2.1a dargestellt [DLL00; Lam01; VA15].

Erhält der Proposer auch hier wieder von der Mehrheit eine *Accepted*-Nachricht, ist die Eingabe entschieden [DLL00]. Um die Entscheidung bekannt zu geben, sendet der Proposer eine *Success*-Nachricht mit der entschiedenen Eingabe v an alle Server, siehe Abbildung 2.1b [DLL00; Lam96]. Die Empfänger werden zu sogenannten Learnern, führen die entschiedene Eingabe aus und antworten dem Proposer mit einem *Acknowledgement* [DLL00; Lam01]. So führt der Proposer das Versenden der *Success*-Nachricht theoretisch so lange aus, bis er von jedem Server eine Bestätigung empfangen hat [DLL00]. Nach Lamport gibt es weitere Möglichkeiten, eine entschiedene Eingabe bekanntzugeben [Lam01].

Eigenschaften

Treten keine Fehler auf, werden maximal $6n$ Nachrichten mit n Servern für eine vollständige Runde verwendet [DLL00]. Andernfalls kann es in der Theorie zu unendlich vielen Nachrichten kommen. Bricht eine Runde ab, in der bereits eine Eingabe entschieden wurde, wird die Eingabe durch *Promise*-Nachrichten der neuen Runde übermittelt [CGR07]. Da bei Empfang von *Promise*- und *Accepted*-Nachrichten stets auf eine Mehrheit gewartet wird, ist durch die Überschneidung sichergestellt, dass zumindest einem Acceptor die Eingabe der damaligen Runde bekannt ist.

Für Konsistenz trotz Abstürzen muss ein Server die zuletzt beantwortete *Prepare*-Nachricht und seine letzte *Accepted*-Nachricht inklusive der akzeptierten Eingabe persistent speichern [DLL00]. Zudem sichert sich ein Proposer seine Rundennummer persistent. Bei f Serverausfällen müssen mindestens $2f + 1$ Server vorhanden sein, um weiterhin einen Fortschritt zu ermöglichen und Mehrheiten bilden zu können [MJ13]. Um eine Runde zu terminieren, darf es zu keiner Unterbrechung durch

Runden anderer Proposer kommen [Lam96]. Kommt es andernfalls zu einem Absturz des Proposers, und kein anderer Server beginnt eine neue Runde, kommt es ebenfalls zu keiner Terminierung des Protokolls.

Des Weiteren zeigten Fischer et al., dass die Möglichkeit besteht, dass es zu keiner Terminierung kommt, wenn bereits ein Server ausfällt [FLP85]. Um Fortschritt in gewissem Maße zu gewährleisten, schlägt Lamport vor, einen einzelnen Proposer, durch Timeouts oder per Zufall, zu wählen, welches ebenfalls einer Art Konsensfindung entspricht [Lam01; Lam06]. Auf eine genauere Implementierung geht er dabei nicht ein. Unabhängig von der Anzahl der Proposer wird die benötigte Konsistenz durchgehend eingehalten [Lam06].

2.2.2 Multi-Paxos

In der Praxis ist es üblich, sich nicht nur auf eine einzelne Eingabe zu einigen, sondern Konsens für mehrere Eingaben zu finden [CGR07; VA15]. Wird der aktuelle Proposer nicht unterbrochen, kann er weitere Runden des Paxosalgorithmus für weitere Eingaben durchführen [DLL00]. Die Anzahl der Nachrichten kann hierbei minimiert werden, indem die *Prepare*- und *Promise*-Nachrichten für alle Eingaben desselben Proposers zu jeweils einer einzelnen zusammengefügt werden. Mit diesen Nachrichtentypen wird ausschließlich die Aktualität einer beginnnenden Runde überprüft. Alle weiteren Nachrichtentypen (*Accept*, *Accepted*, *Success*, *Ack*) müssen für jede Eingabe separat übermittelt werden. Wie in Paxos, können andere Server versuchen, durch höhere Rundennummern Proposer zu werden [CGR07]. Die Korrektheit von Multi-Paxos kann somit aus der Korrektheit von Paxos abgeleitet werden [DLL00].

2.2.3 Konsensusprotokolle für Quantencomputer

Es wurden bereits einige Konsensusprotokolle entwickelt, die die besonderen Eigenschaften von Quantencomputern verwenden. Auf ein paar dieser Protokolle wird im Folgenden eingegangen.

Konsensusprotokoll nach D'Hondt und Panangaden

Ein Protokoll, welches quantenmechanische Eigenschaften verwendet, ist von D'Hondt und Panangaden [DP04]. Der Algorithmus nutzt die Verschränkung der GHZ-Zustände, um einen Konsens zwischen den Prozessen zu finden. Hierbei wird von einem synchronen und anonymen, verteilten System ausgegangen. In diesem Kontext kann bei einer synchronen Umgebung davon ausgegangen werden, dass alles innerhalb eines bestimmten Zeitraums stattfindet [ST17]. So kann zudem festgestellt werden, ob ein Prozess ausgefallen ist. Durch die Anonymität sind die Prozesse identisch und haben keine eigene Identifikation [DP04].

Für den Algorithmus, in dem sich auf den Wert eines einzelnen Qubits geeinigt werden soll, wird bei der Initialisierung ein GHZ-Zustand aus n Qubits erzeugt, siehe Gleichung (2.33) [DP04]. Jeder der n Prozesse erhält jeweils ein Qubit des initialisierten GHZ-Zustands. Nach der Initialisierung speichert sich jeder Prozess sein Qubit. Im zweiten Schritt misst jeder Prozess das Qubit und speichert den gemessenen Wert. Durch die Eigenschaften des GHZ-Zustands, messen alle Prozesse entweder den Wert 0 oder 1. Es wird nicht über vorgeschlagene Eingaben von Extern, wie beispielsweise von

Clients, entschieden. Der Zufall entscheidet über den Wert des Ergebnisses. Für dieses Protokoll wird keine Kommunikation verwendet und hat die Laufzeit $O(1)$ [DP04]. Des Weiteren soll es auch in asynchronen Netzwerken und mit beliebig vielen bösartigen Prozessen funktionieren.

Zudem schlagen D'Hondt und Panangaden einen ähnlich verlaufenden Algorithmus für die Wahl eines Leaders vor [DP04]. Hierfür wird bei der Initialisierung ein W-Zustand, siehe Gleichung (2.34) in Abschnitt 2.1.6, mit n Qubits erzeugt. Auch hier erhält jeder Prozess einen der n Qubits. Anschließend speichert sich wieder jeder Prozess sein Qubit. Daraufhin messen die Prozesse ihr Qubit und speichern das Ergebnis. Ist das Messergebnis bei einem Prozess 1, dann ist er der Leader. Ist es 0, ist er kein Leader. Durch die Eigenschaften des W-Zustands wird immer genau ein Prozess zum Leader gewählt. Auch hier wird keinerlei Kommunikation zwischen den Prozessen benötigt, und die Laufzeit ist $O(1)$. Auch dieses Protokoll soll ebenso im asynchronen Netzwerk funktionieren.

Allerdings kritisieren Gavoille et al. die Algorithmen von D'Hondt und Panangaden [DP04; GKM09]. Zum einen liegt, ihrer Meinung nach, der Vorteil der beiden Protokolle in der Initialisierung und nicht an den quantenmechanischen Eigenschaften [GKM09]. So könnte genauso gut eine klassische zentrale Helferfunktion bei der Initialisierung verwendet werden. Zum anderen behaupten Gavoille et al., dass es keinen Konsens ohne Kommunikation geben kann und die Definition des Problems der Konsensfindung im verteilten System von D'Hondt und Panangaden falsch verstanden wurde [DP04; GKM09]. Denn zu Beginn der eigentlichen Problemstellung sind bereits Eingaben den jeweiligen Prozessen zugeordnet, und als Ziel soll sich auf eine der gegeben Eingaben geeinigt werden [GKM09]. D'Hondt und Panangaden hingegen gehen von keinen gegebenen Eingaben aus [DP04].

Konsensusprotokoll nach Chlebus et al.

Chlebus et al. haben ein Konsensusprotokoll entwickelt, bei dem die Prozesse initial die Eingabe 0 oder 1 haben [CKS10]. Es handelt sich um einen binären Konsensus. Auch hier wird von einem synchronen, verteilten System ausgegangen. Dabei werden keine verschränkten Qubits vor Ablauf des Protokolls verteilt. Für die Kommunikation werden klassische und Quantenkanäle verwendet. Bei dem Fehlermodell des Konsensusprotokolls für Quantencomputer wird von einem Widersacher ausgegangen, der steuern kann, wann die einzelnen Prozesse ausfallen [CKS10]. Zudem sieht dieser alle Inhalte der Qubits und der ausgetauschten Nachrichten. Trotz des Widersachers soll einer der Algorithmen von Chlebus et al. mit Wahrscheinlichkeit $\frac{1}{2}$ einen Konsens finden bei $t < \frac{n}{3}$ und einer Laufzeit von $O(\log n)$. Wobei t die Anzahl der Ausfälle ist und n die Anzahl der Prozesse.

Im ersten Schritt wird die bestehende Eingabe eines Prozesses in einem Qubit gespeichert [CKS10]. Dieses Qubit dient als Kandidat bei der Abstimmung. Im zweiten Schritt wird eine Ticketnummer erzeugt, die eine Superposition aus den Zahlen zwischen 1 und n^3 ist. Anschließend werden innerhalb einer Schleife zunächst Nachrichten erzeugt, die mit der Ticketnummer und dem Kandidaten verschränkt ist. Diese werden an die benachbarten Prozesse gesendet. Bei jeder erhaltenen Nachricht werden die neue und die bestehende Ticketnummer verglichen, und der Kandidat der größeren Ticketnummer gespeichert. Nach Beenden der Schleife werden die Ticketnummer und der Kandidat gemessen. Und der Prozess entscheidet sich schließlich für den Kandidaten. Hat der Prozess hingegen weniger Nachrichten erhalten als festgelegt, fragt dieser bei den benachbarten Prozessen nach deren Kandidaten. Durch das Versenden von mit dem Sender verschränkten

Nachrichten, entsteht eine Verschränkung zwischen Sender und Empfänger [CKS10]. So wird eine globale Verschränkung erzeugt. Mit der anschließenden Messung erhalten die Prozesse dieselben Messergebnisse, vorausgesetzt sie sind in derselben Nachbarschaft. Durch die Superposition der übertragenen Qubits für die Ticketnummern kann der Widersacher nicht auf die späteren Messergebnisse schließen.

Für das Protokoll haben Chlebus et al. drei Quantengatter entwickelt [CKS10]. Zum einen das Controlled-Swap, um die Ticketnummern und den Kandidaten bei der Vergleichsoperation austauschen zu können. Aufgebaut ist ein Swap-Gatter aus einer Reihe an CNOT-Gattern, siehe Definition (2.25) aus Abschnitt 2.1.5. Für den Tausch zweier Qubits, werden drei CNOT-Gatter benötigt. Das erste CNOT hat das erste Qubit als *Control*-Qubit. Das zweite CNOT das zweite als *Control*-Qubit. Das dritte hat erneut das erste als *Control*-Qubit. Für beliebige Quantenregister gleicher Größe wird so bei Anwenden von Swap aus AB schließlich BA. Dieses Gatter kann erweitert und mit einem Qubit $|x\rangle$ gesteuert werden. So wird mit Controlled-Swap$(|x\rangle, A, B)$ schließlich $\alpha|0\rangle AB + \beta|1\rangle BA$ erzeugt, mit $|x\rangle = \alpha|0\rangle + \beta|1\rangle$. Ist $|x\rangle = |1\rangle$, wird A und B getauscht.

Ein weiterer Operator ist Entangled-Copy, welcher ermöglicht, eine zum Original verschränkte Replikation eines Quantenregisters zu erzeugen, ohne das No-Cloning-Theorem zu verletzen, siehe Abschnitt 2.1.5 [CKS10]. Sind zwei Quantenregister A und B gegeben. Alle Qubits von B sind im Zustand $|0\rangle$ und A ist das zu kopierende Register. Durch den Operator werden die einzelnen Qubits an der jeweils gleichen Stelle in den Registern paarweise mit einem CNOT verknüpft. Die Werte von A werden so in B kopiert. Die Qubits von A sind nun mit den Qubits von B verschränkt. Verwendet wird dieser Operator, um die Nachrichten im Algorithmus erzeugen zu können.

Der dritte Operator von Chlebus et al. ist Conditional-Not$(F(A, B), C)$, welcher benutzt wird, um die Ticketnummern miteinander zu vergleichen und daraufhin den Austausch dieser, wenn nötig, auszulösen [CKS10]. Hierbei wird eine boolsche Funktion F für die Bedingung $A > B$ verwendet, wobei auch hier A und B beliebige, gleich große Quantenregister sind. Ist die Bedingung erfüllt, wird eine Negation auf einem 1-Qubit-Register C ausgeführt. Wie ein Vergleich zwischen zwei Superpositionen realisiert wird, ist von Chlebus et al. nicht näher erläutert.

Konsensusprotokoll nach Ben-Or und Hassidim

Ben-Or und Hassidim haben unter anderem ein vereinfachtes Konsensusprotokoll entwickelt, welches auf einem klassischen Protokoll von Chor et al. aufbaut und vor allem die Vorteile quantenmechanischer Eigenschaften nutzt und vorzeigen soll [BH05; CMS89]. Hierbei handelt es sich um ein randomisiertes Konsensusprotokoll, in dem sich die Prozesse auf ein zufälliges, globales Bit in konstanter Zeit einigen [Asp02; BH05]. Durch den Zufall ist die Wahrscheinlichkeit des Nichtterminierens einer Konsensfindung nahe Null [Asp02]. Für die Kommunikation werden für das Protokoll Quantenkanäle verwendet [BH05]. Es ist für Widersacher entwickelt, die Zustände und Nachrichten lesen können und die Prozesse ausfallen lassen, indem sie stoppen. Hierbei wird von einer synchronen Umgebung ausgegangen.

Die Idee besteht darin, dass im ersten Schritt jeder Prozess einen GHZ-Zustand mit n Qubits erzeugt, wobei n der Anzahl der Prozesse entspricht [BH05]. Dem jeweiligen Prozess wird das jeweilige Qubit übermittelt. Im zweiten Schritt erzeugt jeder Prozess n-mal die selbe gleichverteilte Superposition aus allen Zahlen von 1 bis n^3. Jeder Prozess erhält eine dieser Superpositionen. Daraufhin misst jeder Prozess alle im zweiten Schritt erhaltenen Superpositionen. Der Prozess,

aus dessen Superposition das höchste Messergebnis hervorgeht, ist der Leader der Runde. Als nächstes wird das von ihm erhaltene Qubit des GHZ-Zustands gemessen. Dies ist schließlich das Ergebnis des Protokolls. Auch hier erhält der Widersacher durch die Superpositionen keine weiteren Informationen darüber, welcher Prozess der Leader ist, um diesen gezielt abstürzen zu lassen. So geschieht der Ausfall des zukünftigen Leaders per Zufall, mit einer Wahrscheinlichkeit kleiner $\frac{1}{3}$. Auch dass mehrere Prozesse den höchsten Messwert erlangen, und so entschieden werden muss, welcher schließlich der einzige Leader ist, ist durch den großen Zahlenbereich sehr unwahrscheinlich.

Das Protokoll hält $t < \frac{n}{3}$ Fehlern stand [BH05]. Die Wahrscheinlichkeit, dass es sich bei dem Ergebnis des Protokolls tatsächlich um einen initialen Eingabewert handelt, ist bei einem zufälligen Konsens auf 0 oder 1 bei 50%. Dass sich alle auf diese Eingabe einigen, liegt bei einer Wahrscheinlichkeit von $\frac{1}{3}$. Andererseits könnte auch hier das Argument von Gavoille et al. greifen, da sich im weitesten Sinne auch hier auf eine zufällige und nicht eine initiale Eingabe geeinigt wird wie bei dem Protokoll von D'Hondt und Panangaden [DP04; GKM09]. Allerdings äußern sich Ben-Or und Hassidim dahingehend, dieses Protokoll nicht allzu ernst zu nehmen [BH05].

2.3 Commit-Protokolle

Ein verteiltes Datenbanksystem kann zuverlässiger sein als ein zentrales Datenbanksystem, da einzelne Ausfälle toleriert werden können [BHG87]. Dabei ist in einem verteilten Datenbanksystem ausschlaggebend darauf zu achten, dass Transaktionen, die auf mehr als einem Prozess stattfinden, konsistent ausgeführt werden [ML83]. Hierbei sind Transaktionen atomare Operationen, die entweder vollständig („Commit“) oder gar nicht ausgeführt werden („Abort“) [Ske81; ST17]. Neben der Atomarität unterliegen Transaktionen noch den drei Eigenschaften Konsistenz, Isolation und Dauerhaftigkeit, aus dem Englischen auch kurz ACID (Atomicity, Consistency, Isolation, Durability) [ST17]. Konsistenz bedingt, dass eine Transaktion die Konsistenz der Datenbank bewahrt [BN09]. Isolation steht dafür, dass nebenläufige Transaktionen keinen Einfluss aufeinander haben [ST17]. Dauerhaftigkeit bedingt, dass durchgeführte Transaktionen persistent gespeichert werden.

Eine Schwierigkeit in einem verteilten Datenbanksystem ist, die Anforderung der Atomarität einer Transaktion nicht zu verletzen [BHG87; Ske81]. So muss eine Transaktion entweder auf allen involvierten Prozessen ausgeführt und sichtbar werden oder auf keinem der Prozesse [ML83; ST17]. Ebenso dürfen Ausfälle oder Kommunikationsverluste diese Anforderung nicht beeinflussen [ML83]. Um diese Konsistenz trotz Fehlern zu bewahren und ein einheitliches Commit oder Abort einer Transaktion zu erhalten, werden sogenannte Commit-Protokolle verwendet [BHG87; ML83]. So muss auch hier eine Art Konsens gefunden werden.

In den folgenden Abschnitten wird zunächst das 2-Phasen-Commit-Protokoll [Gra78] und ein kooperatives Terminierungsprotokoll vorgestellt, siehe Abschnitt 2.3.1 und 2.3.2. Anschließend wird das 3-Phasen-Commit-Protokoll [Ske81] und ein Terminierungsprotokoll in den Abschnitten 2.3.3 und 2.3.4 präsentiert.

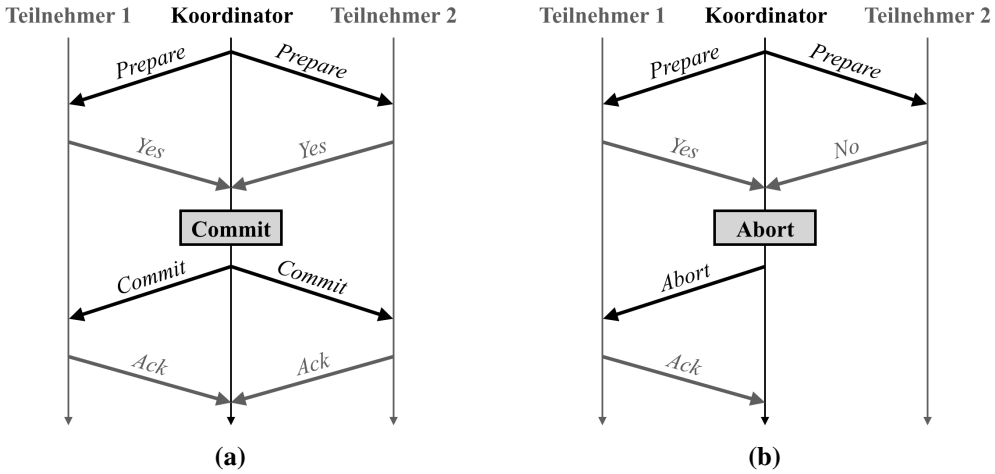

Abbildung 2.2: Kommunikationsverlauf des 2-Phasen-Commit-Protokolls, orientiert an [BN09; Dür18]. (a) Commit einer Transaktion. (b) Abort einer Transaktion.

2.3.1 2-Phasen-Commit-Protokoll

Das bekannteste Commit-Protokoll ist das 2-Phasen-Commit-Protokoll von Gray, dargestellt in Abbildung 2.2 [BHG87; Gra78]. Vereinfacht gesehen ist der Prozess, der zuständig für die jeweilige Transaktion ist, der Koordinator der Transaktion [BHG87]. Die anderen Prozesse sind Teilnehmer. Dabei sind dem Koordinator alle Teilnehmer bekannt. Umgekehrt kennen die Teilnehmer den Koordinator. Untereinander müssen sich die Teilnehmer jedoch nicht kennen. Des Weiteren sind sowohl Transaktionen als auch Prozesse mit einzigartigen Namen gekennzeichnet [ML83]. Von byzantinischen Fehlern wird nicht ausgegangen, die Prozesse fallen durch Halten aus [BHG87; Zha07]. Zudem werden Timeouts verwendet, um Ausfälle erkennen oder zumindest annehmen zu können [BHG87; ST17]. Daher wird zumindest angenommen, dass die Umgebung teilweise synchron ist und es so eine obere, zeitliche Grenze für die Nachrichtenübermittlung gibt.

Ablauf

Wenn eine Transaktion mit einem Commit abgeschlossen werden soll, startet das 2-Phasen-Commit-Protokoll, indem der Koordinator im ersten Schritt *Prepare*-Nachrichten an alle Teilnehmer sendet, siehe den Kommunikationsverlauf in Abbildung 2.2 und den Zustandsautomat des Koordinators in Abbildung 2.3a [BHG87; ML83]. Empfängt ein Teilnehmer das *Prepare*, antwortet dieser mit einem *Yes*, wenn er bereit für ein Commit der Transaktion ist, wie in Abbildung 2.2a dargestellt [BHG87; ST17]. Nach Senden der Antwort ist der Teilnehmer blockiert und wartet auf das Ergebnis des Koordinators, wie in dem Zustandsautomat in Abbildung 2.3b gezeigt [ML83]. Ist dies nicht der Fall, und die Transaktion soll abgebrochen werden (Abort), antwortet der Teilnehmer mit einem *No*, wie bei Teilnehmer 2 in Abbildung 2.2b zu sehen [BHG87; ST17]. Daraufhin schließt der Teilnehmer die Transaktion mit einem Abort ab, wie in Abbildung 2.3b dargestellt, und wartet nicht weiter auf den Koordinator [ML83]. Die Abstimmungsphase ist beendet [ST17]. Der Koordinator wartet auf alle Antworten.

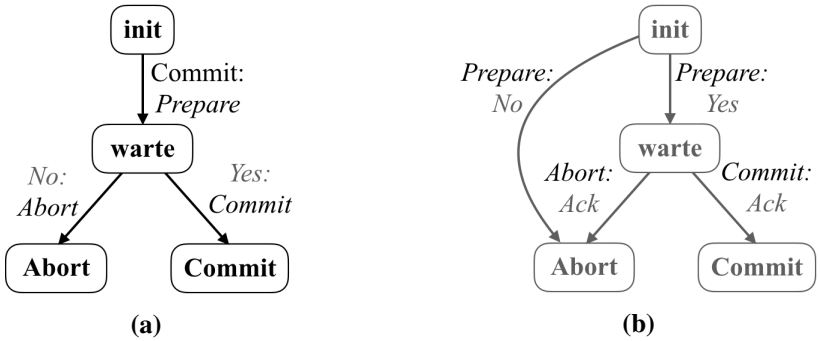

Abbildung 2.3: Zustandsautomaten des 2-Phasen-Commit-Protokolls, orientiert an [ST17]. (a) Zustandsautomat des Koordinators. (b) Zustandsautomat der Teilnehmer.

Haben nun alle Teilnehmer mit einem *Yes* gestimmt, entscheidet der Koordinator, wie in den Abbildungen 2.2a und 2.3a zu sehen, das Commit für die Transaktion und gibt das Ergebnis mittels *Commit*-Nachricht an alle bekannt [ML83; ST17]. Hat jedoch mindestens ein Teilnehmer mit einem *No* geantwortet, entscheidet der Koordinator das Abort für die Transaktion, dargestellt in den Abbildungen 2.2b und 2.3a. Daraufhin sendet er an alle Teilnehmer, die mit einem *Yes* antworteten, eine *Abort*-Nachricht [ML83]. Bei Empfang einer *Commit*-Nachricht, entscheiden die Teilnehmer ebenfalls das Commit für die Transaktion, antworten dem Koordinator mit einer *Ack*-Nachricht und schließen die Transaktion ab, siehe die Abbildungen 2.2a und 2.3b. Wird eine *Abort*-Nachricht empfangen, entscheiden die Teilnehmer das Abort für die Transaktion, antworten ebenfalls mit einer *Ack*-Nachricht und schließen die Transaktion ab, zu sehen in Abbildung 2.2b und dem Zustandsautomat in Abbildung 2.3b. Nun ist die Entscheidungsphase beendet [ST17]. Hat der Koordinator von jedem Teilnehmer, auf dessen Antwort er wartet, eine Bestätigung (*Ack*) erhalten, schließt auch er die Transaktion ab [ML83].

Timeouts

Um durch Ausfälle anderer Prozesse ein möglicherweise unbegrenztes Warten auf eine Nachricht zu vermeiden, werden Timeouts verwendet [ST17]. Erhält ein Teilnehmer bereits zu Beginn des Protokolls keine *Prepare*-Nachricht vor einem Timeout, entscheidet er das Abort, sendet ein *No* an den Koordinator und schließt die Transaktion ab [BHG87; ST17]. Empfängt der Koordinator bis zu einem Timeout, nach Senden der *Prepare*-Nachricht, nicht von allen Teilnehmern ein *Yes* oder *No*, entscheidet er das Abort der Transaktion und sendet *Abort* an alle [ST17]. Hat jedoch ein Teilnehmer ein *Yes* gesendet, und der Timeout für das Empfangen der Entscheidung des Koordinators läuft ab, kann der Teilnehmer selbst keine Entscheidung treffen, ohne die Konsistenz zu gefährden [BHG87]. In diesem Fall bleibt der Teilnehmer blockiert.

Wiederherstellung

Nach jedem Schritt speichern sich die Prozesse ihren aktuellen Zustand persistent, um ein Wiederherstellen nach einem Ausfall zu ermöglichen [ST17]. Wenn ein Teilnehmer abstürzt, während er in Zustand „init“ auf die *Prepare*-Nachricht, siehe Abbildungen 2.2 und 2.3b, des Koordinators wartet, kann er nach einer Wiederherstellung die Transaktion mit einem Abort abschließen und

benachrichtigt den Koordinator dementsprechend. Fällt ein Teilnehmer aus, als die Transaktion bereits bei ihm entschieden ist, wird der jeweilige Zustand „Commit“ oder „Abort“ in Abbildung 2.3b wiederhergestellt, und er gibt das Commit oder Abort erneut dem Koordinator bekannt. Hat ein Teilnehmer direkt vor seinem Absturz mit einem *Yes*, wie in Abbildung 2.2 dargestellt, gestimmt, muss er ohne weitere Hilfsmittel auf die Antwort des Koordinators warten, siehe Zustand „warte“ in Abbildung 2.3b [BHG87]

Stürzt der Koordinator ab, nachdem er *Prepare*-Nachrichten versendet hat, muss er nach der Wiederherstellung in den Zustand „warte“ in Abbildung 2.3a die *Prepare*-Nachrichten erneut versenden, um alle Stimmabgaben der Teilnehmer zu erhalten [ST17]. Wenn der Koordinator bereits vor einem Absturz ein Commit oder Abort entschieden hat, kann er bei der Wiederherstellung in den jeweiligen Zustand „Commit“ oder „Abort“ zurückkehren und die Entscheidung den Teilnehmern bekanntgeben.

Kommunikationsaufwand

Ohne Fehler benötigt ein Durchlauf des 2-Phasen-Commit-Protokolls $4n$ Nachrichten [BHG87; ML83]. Wobei n die Anzahl der Teilnehmer ist [BHG87]. Die Bestätigungsnachrichten *Ack* sind hierbei miteinbezogen. Mit Koordinator sind es insgesamt $n + 1$ Prozesse.

2.3.2 Kooperatives Terminierungsprotokoll für das 2-Phasen-Commit-Protokoll

Um die Blockade eines Teilnehmers im Zustand „warte“ in Abbildung 2.3b, verursacht durch Ablauf des Timeouts oder nach einer Wiederherstellung, auflösen zu können, wird ein Terminierungsprotokoll verwendet [BHG87]. Hierbei ermöglicht das *Cooperative Termination Protocol* den Austausch zwischen den Teilnehmern. Haben andere die Entscheidung des Koordinators erhalten, kann der Teilnehmer diese übernehmen [BN09; ST17]. Damit die Teilnehmer sich untereinander austauschen können, übergibt der Koordinator mit dem Senden der *Prepare*-Nachricht jeweils eine Liste mit allen Teilnehmern [BHG87].

Kommt es nun bei einem Teilnehmer zu einem Timeout oder einer Wiederherstellung nachdem er mit *Yes* in der Abstimmungsphase geantwortet hat, sendet dieser als Nachfrage ein *Decision-Request* an alle anderen Teilnehmer [BHG87; BN09]. Bei Erhalt der Nachfrage antwortet ein Empfänger wie folgt:

(a) Wenn er von dem Koordinator ein *Commit* oder *Abort* erhalten hat, antwortet er dem fragenden Teilnehmer ebenfalls mit einem *Commit* oder *Abort* [BN09].

(b) Wenn er selbst noch kein *Yes* oder *No* an den Koordinator übermittelt hat, schließt er die Transaktion mit einem Abort ab und sendet dem Teilnehmer ein *Abort* [BHG87]. In diesem Fall kann der Koordinator wegen fehlender Stimmen noch kein Commit entschieden haben [BN09].

(c) Wenn er ebenfalls ein *Yes* gesendet, aber selbst noch keine Entscheidung des Koordinators empfangen hat, antwortet er dem fragenden Teilnehmer mit *Uncertain* und beide bleiben blockiert [BHG87; BN09].

Sobald der nachfragende Teilnehmer eine Antwort der Art (a) oder (b) erhält, entscheidet er die Transaktion entsprechend mit einem Commit oder Abort [BHG87]. Mit Antworten, die nur (c) entsprechen, bleibt die Blockade ungelöst, und es muss auf die Wiederherstellung des Koordinators gewartet werden. Da selbst das kooperative Terminierungsprotokoll solch eine Blockade nicht auflösen kann, wird das 2-Phasen-Commit-Protokoll aus Abschnitt 2.3.1 auch blockierendes Commit-Protokoll genannt [BHG87; ST17].

Für den Fall, dass jeder Teilnehmer das Terminierungsprotokoll ausführt, werden $\frac{n(3n+1)}{2}$ benötigt [BHG87]. Mit den $4n$ Nachrichten des 2-Phasen-Commit-Protokoll addiert, ergeben sich daraus $\frac{n(3n+9)}{2}$ Nachrichten.

2.3.3 3-Phasen-Commit-Protokoll

Skeen entwickelte 1981 aufgrund der blockierenden Eigenschaften des 2-Phasen-Commit-Protokolls das 3-Phasen-Commit-Protokoll [Ske81]. In seiner Version werden nur Ausfälle durch Halten der Prozesse angenommen, keine Kommunikationsprobleme [Ske81; ST17]. Des Weiteren befinden sich die Prozesse in einer synchronen Umgebung, sodass Ausfälle mittels Timeouts festgestellt werden können [Ske81]. Unter diesen Bedingungen ist das Protokoll nicht-blockierend solange nicht alle Prozesse ausgefallen sind [BHG87; ST17]. Verwendet wird das 3-Phasen-Commit-Protokoll selten [ST17]. In der Praxis tauchen die blockierenden Fälle des 2-Phasen-Commit-Protokolls, siehe Abschnitt 2.3.1, nur mit geringer Häufigkeit auf. Ebenso wird im 3-Phasen-Commit-Protokoll von einem Koordinator und mehreren Teilnehmern ausgegangen, die bei einer verteilte Transaktion involviert sind. Während des eigentlichen Protokolls werden nur Nachrichten zwischen dem Koordinator und den Teilnehmern ausgetauscht [Ske81]. Für bestimmte Fälle tauschen die Teilnehmer untereinander Nachrichten aus [BHG87].

Es gibt zudem eine Variante des 3-Phasen-Commit-Protokolls, die neben Prozessausfällen auch Kommunikationsfehler toleriert [BHG87]. Diese Variante ist allerdings blockierend, sobald, selbst ohne Auftreten von Kommunikationsfehlern, die Hälfte der Prozesse ausgefallen ist.

Ablauf

Zu Beginn des Protokolls sendet der Koordinator an jeden Teilnehmer eine *Prepare*-Nachricht, siehe den Kommunikationsverlauf in Abbildung 2.4 [BHG87; ST17]. Daraufhin antworten diese ihm entweder mit einem *Yes* und stimmen für einen Commit oder *No* und stimmen für einen Abort der Transaktion. Nach der Abstimmung mit einem *No* schließt der Teilnehmer die Transaktion mit einem Abort ab, wie bei dem 2-Phasen-Commit-Protokoll in Abschnitt 2.3.1, siehe den Zustandsautomat in Abbildung 2.5b [BHG87]. Ist mindestens eine Stimme nach Sammeln der Antworten ein *No*, entscheidet der Koordinator die Transaktion mit einem Abort und sendet *Abort*-Nachrichten an alle Teilnehmer, die für ein Commit waren, siehe die Abbildungen 2.4b und 2.5a. Diese warten auf die Antwort des Koordinators, entscheiden bei Empfang der *Abort*-Nachricht ebenfalls das Abort der Transaktion und schließen sie ab, wie in dem Zustandsautomat in Abbildung 2.5b dargestellt.

Enthalten hingegen alle Stimmen aller Teilnehmer ein *Yes*, antwortet der Koordinator den Teilnehmern mit *Pre-Commit* [BHG87]. Hierin unterscheidet sich das 3-Phasen-Commit-Protokoll, siehe die Abbildungen 2.4a und 2.5. Empfangen die Teilnehmer eine *Pre-Commit*-Nachricht, bestätigen sie deren Empfang mit einem *Pre-Commit-Ack* [BHG87; ST17].

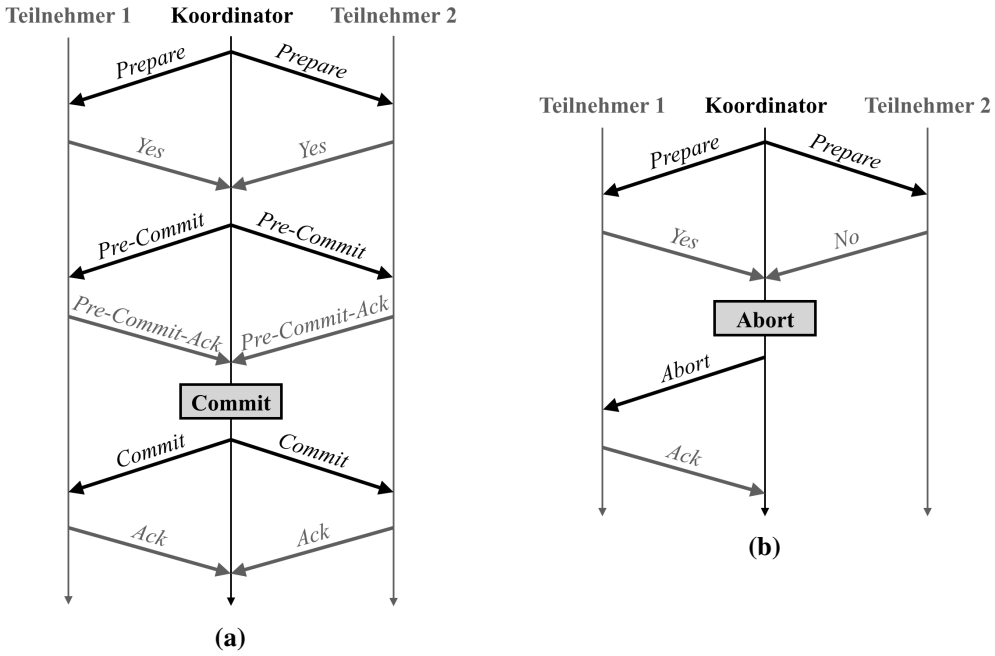

Abbildung 2.4: Kommunikationsverlauf des 3-Phasen-Commit-Protokolls, orientiert an [BHG87; BN09; Dür18; ST17]. (a) Commit einer Transaktion. (b) Abort einer Transaktion, wie bei 2-Phasen-Commit-Protokoll, Abschnitt 2.3.1.

Empfängt der Koordinator von allen Teilnehmern ein *Pre-Commit-Ack*, kann die Transaktion mit einem Commit entschieden werden [BHG87; ST17]. Die Entscheidung wird schließlich, wie in Abbildung 2.4a dargestellt, mit *Commit*-Nachrichten an alle bekannt gegeben. Die Teilnehmer warten erneut auf die Antwort des Koordinators und können bei Erhalt die Transaktion mit einem Commit abschließen, zu sehen in Abbildung 2.5b [BHG87]. Den Erhalt der Nachricht bestätigen sie ihm erneut mit einem *Ack*, sodass auch der Koordinator die Transaktion abschließen kann [ST17].

Timeouts

Erhält zu Beginn des Protokolls ein Teilnehmer bis zu einem Timeout keine *Prepare*-Nachricht, wird die Transaktion von ihm mit einem Abort abgeschlossen und benachrichtigt den Koordinator über seine Entscheidung [ST17]. Empfängt ebenso der Koordinator bis zu einem Timeout nicht von allen eine Antwort auf seine *Prepare*-Nachricht, wird die Transaktion mit einem Abort entschieden, daraufhin sendet er an alle *Abort*-Nachrichten.

Befindet sich der Koordinator im Zustand „Pre-Commit“, siehe Abbildung 2.5a, und der Timeout läuft ab, bevor er von allen ein *Pre-Commit-Ack* empfangen hat, entscheidet er ein Commit und gibt dies mittels *Commit*-Nachrichten bekannt [BHG87; ST17]. Die Teilnehmer haben zuvor alle mit *Yes* für ein Commit gestimmt, sodass ausgefallene Teilnehmer nach einer Wiederherstellung weiterhin für ein Commit sind [ST17]. Worin sich das 3-Phasen-Commit-Protokoll vom 2-Phasen-Commit-Protokoll unterscheidet.

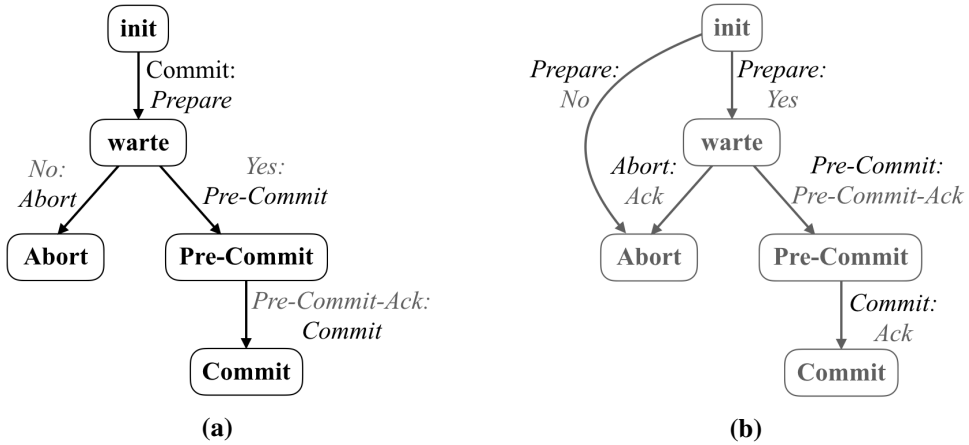

Abbildung 2.5: Zustandsautomaten des 3-Phasen-Commit-Protokolls, orientiert an [ST17]. (a) Zustandsautomat des Koordinators. (b) Zustandsautomat der Teilnehmer.

Hat ein Teilnehmer, während er sich im Zustand „warte“, siehe Abbildung 2.5b, befindet, einen Timeout, wird ein Terminierungsprotokoll hinzugezogen [BHG87]. Dasselbe gilt für einen Teilnehmer, der im Zustand „Pre-Commit“ einen Timeout hat, während er auf das *Commit* des Koordinators wartet. Der Teilnehmer muss sicherstellen, dass auch die anderen Teilnehmer sich im selben Zustand wie er befinden, um ein Commit zu entscheiden. Es darf kein anderer Teilnehmer mehr im unsicheren Zustand „warte“ sein.

Wiederherstellung

Wie im 2-Phasen-Commit-Protokoll, wird auch im 3-Phasen-Commit-Protokoll der aktuelle Zustand eines Prozesses für eine Wiederherstellung nach Ausfall persistent gespeichert [BHG87; ST17]. Wird ein Teilnehmer im Zustand „init“ in Abbildung 2.5b wiederhergestellt, entscheidet er die Transaktion mit einem Abort und benachrichtigt den Koordinator [BHG87]. Fällt ein Teilnehmer in den Zuständen „Commit“ und „Abort“ aus, befindet er sich auch nach Wiederherstellung wieder in diesen, wie im 2-Phasen-Commit-Protokoll Abschnitt 2.3.1. Befindet sich ein Teilnehmer im Zustand „warte“ oder „Pre-Commit“, siehe Abbildung 2.5b, als er ausfällt, muss er sich mit den anderen Teilnehmern austauschen, um den aktuellen Stand des Protokolls ermitteln zu können.

Stürzt der Koordinator ab, funktioniert die Wiederherstellung wie bei dem 2-Phasen-Commit-Protokoll in Abschnitt 2.3.1. Nach der Wiederherstellung in den jeweiligen Zustand („warte“, „Abort“, „Pre-Commit“ und „Commit“ in Abbildung 2.5a) versendet er die jeweils zugehörigen Nachrichten (*Prepare*, *Abort*, *Pre-Commit*, *Commit*) erneut [BHG87; ST17].

Kommunikationsaufwand

Läuft das Protokoll ohne Ausfälle durch, werden insgesamt $6n$ Nachrichten, inklusive der Bestätigungen (*Ack*) am Ende, benötigt [BHG87]. Hierbei entspricht auch n erneut der Anzahl der Teilnehmer, ohne Koordinator. So ist der Kommunikationsaufwand, verglichen mit dem 2-Phasen-Commit-Protokoll, durch die zusätzlichen *Pre-Commit*-Nachrichten und den zugehörigen Antworten höher.

2.3.4 Terminierungsprotokoll für das 3-Phasen-Commit-Protokoll

Um Blockaden nach Timeouts der Teilnehmer in den Zuständen „warte“ und „Pre-Commit“ in Abbildung 2.5b zu vermeiden, wird ebenfalls ein Terminierungsprotokoll angewendet [Ske81; ST17]. Dabei wird das Terminierungsprotokoll, welches von Skeen vorgeschlagen wird, verwendet [Ske81]. Ein neuer Koordinator wird gewählt, welcher schließlich den Entscheidungsprozess zu Ende führen soll.

Hat ein Teilnehmer während des Wartens auf eine Antwort des Koordinators im Zustand „warte“ oder „Pre-Commit“ einen Timeout, kann nach den Voraussetzungen in Abschnitt 2.3.3 davon ausgegangen werden, dass der Koordinator ausgefallen ist [BHG87]. Daraufhin wird ein neuer Koordinator gewählt. Dies kann mittels Abstimmungsverfahren oder fester Ordnung der Prozesse anhand deren Identifikationsnummer geschehen [BHG87; Ske81]. Ist der neu ernannte Koordinator bereits selbst im Zustand „Commit“ oder „Abort“, gibt er seine Entscheidung an alle Teilnehmer, wie in Abbildung 2.4 dargestellt, bekannt, und das Terminierungsprotokoll ist beendet [Ske81]. Ist dies nicht der Fall, sendet er schließlich *State-Request*-Nachrichten an alle Teilnehmer [BHG87]. Daraufhin antworten ihm die Teilnehmer mit ihren aktuellen Zuständen, dargestellt in Abbildung 2.5b. Der Koordinator sammelt die Nachrichten und entscheidet anhand dessen:

(a) Befindet sich mindestens ein Teilnehmer in Zustand „Abort“, entscheidet auch er das Abort der Transaktion und gibt die Entscheidung an alle bekannt [BHG87; Ske81].

(b) Befindet sich mindestens ein Teilnehmer in Zustand „Commit“, entscheidet auch er das Commit der Transaktion und gibt die Entscheidung an alle bekannt [BHG87; Ske81].

(c) Befinden sich alle antwortenden Teilnehmer in Zustand „warte“, entscheidet er ein Abort [BHG87].

(d) Befinden sich mindestens ein antwortenden Teilnehmer in Zustand „Pre-Commit“, jedoch keiner in Zustand „Commit“, sendet der Koordinator zunächst *Pre-Commit*-Nachrichten an alle Teilnehmer in Zustand „warte“ [BHG87]. Erhält der Koordinator von ihnen ein *Pre-Commit-Ack*, entscheidet er das Commit, gibt dies allen bekannt und schließt die Transaktion ab.

Trifft der Fall (c) zu, können nicht antwortende Teilnehmer sich höchstens in Zustand „Pre-Commit“ befinden [BHG87]. Allerdings nicht in Zustand „Commit“, da hierfür alle Teilnehmer mindestens in Zustand „Pre-Commit“ sein müssen. Teilnehmer, die während des Terminierungsprotokolls nicht antworten, werden auch nach Wiederherstellung nicht miteinbezogen. Fällt der gewählte Koordinator aus, wird nach einem Timeout ein neuer gewählt. Dies funktioniert, solange nicht alle Prozesse ausgefallen sind.

Für die Fälle (a), (b) und (c) werden $4(n-1)$ Nachrichten benötigt, mit der Annahme, dass nur der ursprüngliche Koordinator ausfällt [BHG87]. *Ack*-Nachrichten sind miteinbezogen. Für den Fall (d) werden mehr Nachrichten benötigt, abhängig von der Anzahl an Teilnehmern, die sich im Zustand „warte“ befinden. Des Weiteren müssen die Nachrichten berücksichtigt werden, die aufgrund der Wahl eines neuen Koordinators anfallen könnten.

3 Quantencomputing in der Praxis

Erst vor kurzem kam die Nachricht auf, dass Google einen Quantencomputer entwickelt hat, der eine bestimmte Berechnung schneller durchführen kann als ein klassischer Computer [AAB+19; Bro19; Gib19]. Die Überlegenheit der Quantencomputer wurde damit gezeigt [AAB+19; Gib19]. Allerdings ist diese Quantenüberlegenheit nur für ein sehr spezielles Problem gezeigt [Gib19]. Bei dem Problem handelt es sich darum, die Ausgaben eines Quantenzufallsgenerator für Zahlen zu überprüfen [AAB+19; Gib19]. Während von Arute et al. angenommen wird, dass die Berechnung mittels klassischem Supercomputer um die 10.000 Jahre benötigt, braucht ihr Sycamore Chip gerade einmal 200 Sekunden [AAB+19].

Konkurrent IBM bestreitet jedoch Googles Annahme über die lange Laufzeit eines klassischen Computers für das genannte Problem und ist der Meinung, dass dies mit einer anderen, klassischen Herangehensweise in nur zweieinhalb Tagen gelöst werden kann [Gib19; PGN+19]. Auch wenn die Aussage von IBM zutreffen sollte, und der praktische Nutzen des Entwickelten noch offen ist, handelt es sich dennoch um einen Meilenstein im Bereich des Quantencomputings [Gib19].

Im Folgenden werden zunächst die Errungenschaften, die durch den Einsatz des Quantencomputings möglicherweise entstehen, aufgezeigt und anschließend die aktuelle Situation der Technologie wiedergegeben.

3.1 Mögliche Errungenschaften durch Quantencomputing

In Hinblick auf den Bereich des Quantencomputings gibt es viele Gebiete, die durch diese Technologie Unterstützung erfahren könnten [Haa15]. Mit Hilfe von Quantensensoren könnten beispielsweise Erdbeben akkurater vorhergesagt werden. Ebenso können weitere Gebiete aus Physik und Chemie durch den Einsatz von Quantencomputing profitieren [Nat19; RP11]. So könnten komplexe Moleküle und schwarze Löcher näher erforscht werden [Pre18].

Des Weiteren gibt es in der theoretischen Informatik Probleme, die, zumindest bisher, mit klassischen Computern nur schwer lösbar sind [Pre18]. Quantenalgorithmen, wie der bekannte von Shor [Sho94], zeigen, dass solche Probleme effizient von Quantencomputern gelöst werden könnten, siehe Abschnitt 2.1.12. Dennoch wird davon ausgegangen, dass nicht alle schweren Probleme durch Quantencomputer effizient lösbar werden.

Zudem könnte in Zukunft durch die Verwendung von Quantenkryptographie die Übertragung von Informationen erheblich sicherer gestaltet werden [Haa15]. Klassische Kryptosysteme wie RSA [RSA78] würden aufgrund des bereits genannten Algorithmus von Shor [Sho94] mit dem Bau großer Quantencomputer hinfällig werden. Auch das aktuelle Themengebiet Machine Learning könnte in Kombination mit Quantencomputing einen erheblichen Vorteil gegenüber klassischem Machine Learning erfahren [Pre18]. So soll es mit Quantencomputern möglich sein, mit einer viel größeren Datenmenge umgehen zu können [Bro19].

3.2 Heutige Situation

Ob und wann Quantencomputer tatsächlich in den in Abschnitt 3.1 genannten Gebieten angewendet werden können, ist unklar [Bro19]. Bisherige Quantencomputer besitzen nur eine geringe Anzahl an Qubits, da sie nur schwer zu kontrollieren sind [Bro19; RP11]. Googles Sycamore Chip enthält 54 Qubits [AAB+19; Gib19]. Des Weiteren sind die für Berechnungen benötigten Quantengatter aufgrund der Umgebung anfällig für Störungen, siehe Abschnitt 2.1.11 [Haa15]. Um die durch Dekohärenz verursachten Fehler zu beseitigen, werden Fehlerkorrekturmechanismen benötigt. Dazu wird wiederum eine hohe Anzahl an Qubits und Quantengattern benötigt, die momentan nicht verfügbar ist [Pre18]. Um den Algorithmus von Shor [Sho94] zu realisieren und damit aktuelle Kryptosysteme brechen zu können, werden Millionen an Qubits benötigt [Nat19; Pre18]. Die Herstellung eines Quantencomputers mit vollständiger Fehlerkorrektur, der große, praxistaugliche Berechnungen durchführen kann, scheint somit noch weit entfernt [Bro19; Nat19].

Quantencomputer, die bereits in den nächsten Jahren hergestellt werden können, werden auch NISQ-Computer genannt [Pre18]. NISQ steht für *Noisy Intermediate-Scale Quantum* [Pre18] und beschreibt Quantencomputer, die bis zu ein paar hundert, störanfällige Qubits enthalten [Nat19; Pre18]. Störanfällig deshalb, da nicht genug Qubits zur Fehlerkorrektur verfügbar sein werden [Pre18]. Zugänglich werden NISQ-Computer für die meisten Nutzer wohl über die Cloud sein.

Eines der ersten Anwendungsgebiete für solche NISQ-Computer sind Optimierungsprobleme, die mit Hilfe von hybriden Algorithmen, siehe Abschnitt 2.1.10, gelöst werden können [Bro19; Pre18]. Ob hybride Algorithmen tatsächlich effizienter als rein klassische Algorithmen sind, ist noch nicht bewiesen [Pre18]. Hybride Algorithmen mit heutigen NISQ-Computer, angewendet auf realistische Problemmengen, zeigen noch keinen Vorteil gegenüber klassischen Computern [Bro19; GM19].

Wie sich Quantencomputing in Zukunft entwickelt, welche Auswirkungen und welchen praktischen Nutzen es haben wird, ist unklar [Haa15; RP11]. So muss weiterhin Grundlagenforschung betrieben werden, um unter anderem die Fehlerraten von Quantengattern verringern zu können [Haa15; Pre18]. Dennoch ist für weitere intensive Forschung und Entwicklung die finanzielle Unterstützung öffentlicher und privater Investoren ausschlaggebend [Nat19]. Forscher haben Sorge, dass, wenn in absehbarer Zeit kein sinnvoller Anwendungsfall für Quantencomputer entdeckt wird, das Interesse und die finanzielle Unterstützung der Investoren abnimmt [Bro19; Nat19]. Ein „Quantenwinter“ ist die Gefahr [Bro19]. Es wird angenommen, dass die Quantenkryptographie wohl am ehesten einen wirtschaftlichen Nutzen aufzeigen können wird [Haa15].

Dennoch bringt die Forschung im Bereich des Quantencomputings auch Verbesserungen für klassisches Computing hervor [Bro19; RP11]. Selbst wenn klassische Supercomputer besser als Quantencomputer sind, könnten Quantencomputer in Zukunft vermehrt eingesetzt werden, falls sie beispielsweise geringere Kosten verursachen [Pre18].

4 Speedup durch Verschränkung?

In diesem Kapitel wird untersucht, ob Verschränkung die Ursache des Speedups von speziellen Quantenalgorithmen verglichen mit klassischen Algorithmen ist. Beispielhaft dafür ist der Algorithmus Shor [Sho94], der einen exponentiellen Speedup gegenüber bisher bekannten klassischen Algorithmen aufweist, siehe Abschnitt 2.1.12 [Mon16; RP11]. Dazu werden im Folgenden Meinungen und Ergebnisse wissenschaftlicher Arbeiten zusammengetragen, die sich dem Thema des Speedups widmen.

Zunächst wird in Abschnitt 4.1 Superposition an sich als möglicher Grund des Speedups betrachtet. Anschließend wird in Abschnitt 4.2 auf die Rolle der Verschränkung bezüglich des Speedups eingegangen. Dabei wird die Simulierbarkeit von Quantensystemen in Abschnitt 4.2.1 sowie Speedup in Abwesenheit von Verschränkung in Abschnitt 4.2.2 erläutert. Zuletzt werden die Erkenntnisse zusammengefasst, siehe Abschnitt 4.3.

4.1 Bedeutung von Superposition

Der in Abschnitt 2.1.5 beschriebene Quantenparallelismus ist eine Besonderheit des Quantencomputings und ermöglicht, exponentiell viele Werte durch einmaliges Anwenden einer Operation zu verändern [DJ07; NC11]. Grundlage dafür bilden die Eigenschaften der Superposition, siehe Abschnitt 2.1.1 [NC11; RP11].

Allerdings gibt es nach Jozsa und Linden zur quantenmechanischen Superposition ein klassisches Pendant [JL03]. So kann sich beispielsweise eine elastisch vibrierende Schnur, dessen Enden fixiert sind, auch in der klassischen Welt in Superposition befinden. Damit schließen Jozsa und Linden darauf, dass die Superposition selbst nicht der Grund für die besonderen Eigenschaften des Quantencomputings sein kann.

4.2 Bedeutung von Verschränkung

Die andere Besonderheit des Quantencomputing ist Verschränkung, siehe Abschnitt 2.1.6 [DJ07]. Dass der Zustand miteinander verschränkter Qubits nicht als Tensorprodukt der einzelnen Zustände der Qubits darstellbar ist und eine Korrelation zwischen ihnen besteht, ist einzigartig für die quantenmechanische Welt [Hom18; NC11; RP11]. Verschränkung ist in der klassischen Welt nicht anzutreffen [Hom18; NC11; RP00]. Daher wird sie häufig als Ursache für den Speedup der Quantenalgorithmen genannt [RP11]. Ob dem so ist, wird im Folgenden näher betrachtet.

4.2.1 Simulierbarkeit von Quantensystemen

Bereits 1982 stellte Feynman fest, dass das Simulieren quantenmechanischer Vorgänge nur schwer möglich ist [Fey82; JL03]. So scheint es, als würde für die Simulation eines Quantensystems auf einem klassischen Computer exponentiell viele Ressourcen mehr benötigt [Fey82; JL03; Vid03].

Wie in Abschnitt 2.1.6 erläutert, sind die meisten Quantenzustände verschränkt [RP11]. Um Transformationen auf beliebigen Superpositionen aus 2^n Zuständen mittels eines klassischen Computers durchführen zu können, werden exponentiell mehr Ressourcen benötigt als mit einem Quantencomputer, der nur linear viele Ressourcen verwendet [EJ98; JL03; Joz98]. Jozsa und Linden zeigen hingegen, dass in Betrachtung von reinen Zuständen Quantenberechnungen, die keine Verschränkung aufweisen, effizient mit einem klassischen Computer zu simulieren sind [JL03; Joz98].

Des Weiteren zeigen Jozsa und Linden auf, dass ausschließlich bipartite Verschränkung, bestehend zwischen zwei Quantensystemen, wie beispielsweise den zwei Qubits eines Bell-Zustands (Abschnitt 2.1.6), nicht ausreichend für einen exponentiellen Speedup eines Quantenalgorithmus ist und effizient simuliert werden kann [JL03; RP11]. So muss ein Quantenalgorithmus mit reinen Zuständen multipartite Verschränkung aufweisen, um nicht klassisch simulierbar zu sein [JL03]. Dazu darf es keine Beschränkung in der Anzahl der verschränkten Qubits geben. Zu multipartiten Quantensystemen gehören beispielsweise der W-Zustand und der GHZ-Zustand, siehe Abschnitt 2.1.6 [RP11]. Zudem zeigt Vidal, dass wenn das Maß an Verschränkung polynomiell in der Anzahl der Qubits nach oben beschränkt ist, für eine klassische Simulation polynomiell viele Ressourcen benötigt werden [Vid03]. Es ist somit effizient simulierbar [H09; Vid03]. Wobei auch das nur für die Betrachtung von Quantenalgorithmen mit reinen Zuständen gilt. Um einen exponentiellen Speedup in einem Quantenalgorithmus mit reinen Zuständen erreichen zu können, wird ein exponentielles Maß an Verschränkung bezüglich der Anzahl der Eingabequbits benötigt [H09; Vid03].

So erzeugt auch der Algorithmus von Shor [Sho94] Zustände mit unbegrenzter, multipartiter Verschränkung [H09; JL03]. Wodurch die Annahme besteht, dass er nicht klassisch simulierbar ist [H09]. Dennoch ist, wie bereits in Abschnitt 2.1.12 angesprochen, nicht ausgeschlossen, dass ein effizienter klassischer Algorithmus existiert [H09; Mon16; RP11]. Des Weiteren haben Kendon und Munro untersucht, welche Rolle Verschränkung während der Ausführung des Algorithmus von Shor [Sho94] spielt [KM04]. Dabei zeigt sich, dass das Maß an Verschränkung mit der Eingabe zusammenhängt [H09; KM04]. Verschränkung wird bei der Ausführung der Berechnung beiläufig erzeugt, was sich nicht vermeiden lässt [KM04]. Im weitesten Sinne aufgebraucht wird sie jedoch nicht, sodass die Annahme besteht, dass zumindest ein häufiges Anwenden von Verschränkung nicht zu einem exponentiellen Speedup führt.

Für bestimmte Methodiken des Quantencomputings zeigt sich außerdem, dass viele Zustände, die in zu hohem Maße verschränkt sind, keinen Nutzen für deren Berechnungen haben [GFE09; RP11]. Im Allgemeinen werden jedoch die Eigenschaften von multipartiter Verschränkung bisher nur sehr wenig verstanden [RP11].

4.2.2 Speedup ohne Verschränkung

Eine weitere Überlegung ist, dass Speedup mit Hilfe von nicht verschränkten, gemischten Zuständen möglich sein könnte, siehe Abschnitt 2.1.9 [DJ07; JL03]. Dafür zeigen Biham et al., dass mit der Verwendung von pseudo-reinen Zuständen bis zu einem bestimmten Grad an Reinheit keine Verschränkung während einer Berechnung erzeugt wird und dennoch ein geringer Speedup gegenüber klassischen Algorithmen erreicht werden kann [BBKM04; DJ07]. Diese pseudo-reinen Zustände verhalten sich wie reine Zustände, sind jedoch gemischte Zustände, die jeweils aus einer Kombination eines reinen und eines gemischten Zustands bestehen [BBKM04; DJ07; JL03].

Dennoch zeigen Linden und Popescu auf, dass auch mit pseudo-reinen Zuständen viele der Quantenalgorithmen mit exponentiellem Speedup, darunter auch der Algorithmus von Shor [Sho94], Verschränkung benötigen [DJ07; LP01]. Des Weiteren gilt, dass wenn das Maß an Verschränkung für gemischte Zustände hinreichend beschränkt ist, das System effizient von einem klassischen Computer simuliert werden kann [DJ07; Vid03].

Unter anderem zeigt Meyer, dass keine Verschränkung verwendet werden muss, um bereits eine Verbesserung von Quantenalgorithmen, verglichen mit klassischen Algorithmen, vorweisen zu können [Mey00; RP11]. Dabei verändern sie die Quantenalgorithmen so, dass das Tensorprodukt nicht in Betracht gezogen wird [Mey00]. Jedoch fallen hierbei exponentielle Kosten für die physikalische Umsetzung an. Diese Kosten fallen auch bei Lloyds Umsetzung des Algorithmus von Grover [Gro96] ohne Verwendung von Verschränkung an [Llo99]. Dennoch zeigt sich bei seiner Umsetzung ein Speedup gegenüber klassischen Suchalgorithmen für Datenbanken.

4.3 Zusammenfassende Erkenntnisse

Im Allgemeinen herrscht keine eindeutige Meinung und Erkenntnis dazu, wo die Rolle der Verschränkung eingeordnet werden muss [DJ07; H09]. Jozsa und Linden gehen davon aus, dass Verschränkung nicht als ausschlaggebende Ressource für den Speedup gegenüber klassischen Computern gesehen werden sollte [JL03]. Auch wenn sich gezeigt hat, dass Quantenalgorithmen mit reinen Zuständen für einen exponentiellen Speedup ein gewisses Maß an Verschränkung benötigen, siehe Abschnitt 4.2.1 [JL03; RP11]. So sollte, nach Vedral, nicht nur von einer einzelnen Ursache für den Speedup ausgegangen werden [Ved10].

Dabei wird unter anderem auch Quantum Discord als eine mögliche Ursache für den Speedup untersucht [LBAW08; Ved10]. Bei Quantum Discord handelt es sich um ein Maß, das, neben Verschränkung, auch weitere Quantenkorrelationen enthält [LBAW08]. So ist trotz nicht verschränkter, gemischter Zustände immer noch ein Maß an Quantum Discord vorhanden [DSC08; LBAW08]. In Bezug auf reine Zustände entspricht Quantum Discord dem Maß an Verschränkung [LBAW08]. Dadurch besteht die Annahme, dass nicht verschränkte, gemischte Zustände dennoch der klassischen Welt überlegen sind [Ved10].

Ob die Verwendung gemischter Zustände in Berechnungen tatsächlich einen Vorteil ergibt und welche Rolle deren Verschränkung spielt, ist bisher noch nicht geklärt [DJ07]. Es kann allerdings angenommen werden, dass, zumindest bezogen auf pseudo-reine und reine Zustände, Verschränkung für Quantenalgorithmen mit exponentiellem Speedup notwendig, jedoch nicht hinreichend ist [DJ07; H09].

5 Verwendung von Quantencomputing in Konsensusprotokollen

In diesem Kapitel werden die Konzepte des Quantencomputings verwendet, um exemplarisch das Konsensusprotokoll Paxos [Lam98] aus Abschnitt 2.2.1 zu erweitern. Dabei soll untersucht werden, ob sich durch die Kombination des klassischen Protokolls mit Quantencomputing Optimierungen bezüglich des Kommunikationsaufwands oder andere Vorteile im Ablauf ergeben. Paxos wurde unter anderem gewählt, da es sich in einer asynchronen Umgebung befindet. So sind in der Realität die meisten verteilten Systeme in einer asynchronen Umgebung [ST17]. Des Weiteren scheint Paxos in Zusammenhang mit Quantencomputern bisher nur sehr wenig erforscht zu sein.

Im Folgenden werden zwei Erweiterungen des Paxosalgorithmus mit Quantencomputing präsentiert, siehe Abschnitt 5.1 und 5.2. Dabei werden jeweils anschließend die sich ergebenden Vorteile der Erweiterungen aufgezeigt. Darauffolgend werden mögliche Probleme und die Betrachtung byzantinischer Fehler angesprochen. Zuletzt werden Erkenntnisse vorgestellt, die während der Entwicklung der Erweiterungen gewonnen wurden, zu sehen in Abschnitt 5.3.

5.1 Paxos: Proposerwahl mit W-Zustand

Die erste Kombination von Paxos mit Quantencomputing ist eine Erweiterung des ursprünglichen Protokolls. Wie in Abschnitt 2.2.1 der Grundlagen erläutert, darf für die Terminierung des Algorithmus die aktuelle Runde nicht durch andere Runden unterbrochen werden [Lam01]. Dies bedeutet, dass es nur einen einzigen Proposer für den vollständigen Durchgang einer Runde geben darf. Wie Lamport bereits annahm, kann dies mit einer Wahl eines einzelnen Proposers erfüllt werden [Lam06]. Hierfür können die besonderen Eigenschaften der quantenmechanischen Welt herangezogen und mit dem klassischen Konsensusprotokoll kombiniert werden. Verwendet wird der W-Zustand, auf den in Abschnitt 2.1.6 genauer eingegangen wird. Dessen Verschränkungseigenschaften werden genutzt, um einen der Server per Zufallsentscheidung in die Rolle des Proposers zu wählen. Es wird somit nicht wie üblich per Abstimmung durch die Beteiligten entschieden, wer Proposer ist, sondern die Idee von D'Hondt und Panangaden verwendet [DP04].

5.1.1 Umgebung und zusätzliche Mechanismen

Wie im originalen Paxos [Lam98] wird von einer asynchronen Umgebung ausgegangen, bei der die Server durch Halten ausfallen [Lam01; VA15]. Zudem werden bei der klassischen Kommunikation ebenso Verluste und Vervielfachungen angenommen [Lam01]. Eine zentrale Einheit generiert den W-Zustand und verteilt dessen Qubits, sodass diese Einheit für jeden Server einen Quantenkanal in Richtung des Servers benötigt, wie beispielhaft in Abbildung 5.1 mit den grauen Quantenkanälen

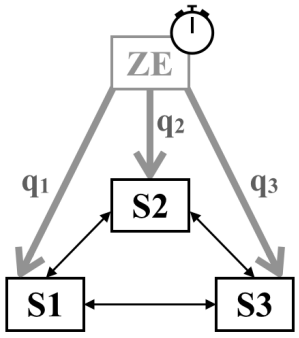

Abbildung 5.1: Kommunikationskanäle des erweiterten Konsensusprotokolls Paxos. Dünne, schwarze Pfeile: klassische Kommunikationskanäle. Dicke, graue Pfeile: Quantenkanäle.

und den Qubits $q1$ bis $q3$ für die Server S1 bis S3 dargestellt. Diese n Quantenkanäle sind zuverlässig, sodass die Qubits ohne Verluste und Veränderungen empfangen werden können. Am Paxosalgorithmus selbst werden keine Kommunikationsveränderung, bezüglich der Nachrichtentypen, vorgenommen. Anhand der schwarzen, bidirektionalen Kanäle in Abbildung 5.1 wird die klassische Kommunikation zwischen den Servern dargestellt.

Die Aufgaben der Generierung und Verteilung könnten auch einem beliebigen teilnehmenden Server zugeteilt werden. Allerdings wäre so eine höhere Anzahl an Quantenkanälen nötig ($n(n-1)$ anstatt n), da angenommen wird, dass Server ausfallen können. So müssten in diesem Fall die Aufgaben auf andere Server übertragbar sein. Die zentrale Einheit hingegen wird als ausfallsicher, beziehungsweise leicht ersetzbar angenommen.

Da die Detektion eines Ausfalls in einem asynchronen Netzwerk, nach Van Renesse und Altinbuken und Fischer et al., nicht möglich ist, löst die zentrale Einheit regelmäßig nach Ablauf eines festgelegten Zeitintervalls eine neue Generierung und Verteilung eines W-Zustands aus, um den Fortschritt des erweiterten Protokolls zu ermöglichen, siehe Abbildung 5.1 [FLP85; VA15]. Dies geschieht solange, bis das Protokoll erfolgreich durch Konsensfindung abgeschlossen ist.

5.1.2 Initialisierungsphase des erweiterten Paxos

Zunächst wird, wie bei D'Hondt und Panangaden, in der Initialisierungsphase ein W-Zustand erzeugt [DP04]. Dieser enthält n verschränkte Qubits, welche den n Servern, die während des Algorithmus beteiligt sind, entsprechen:

$$|W_n\rangle = \frac{1}{\sqrt{n}}(\overbrace{|\underbrace{0...001}_{n\text{ Qubits}}\rangle + |0...010\rangle + ... + |1...000\rangle}^{n\text{ Zustände}}) \qquad ,n : \#\text{ Server} \tag{5.1}$$

Die n möglichen Zustände befinden sich bezüglich der Standardbasis in Superposition. Anschließend werden die einzelnen verschränkten Qubits des W-Zustands, wie in Abbildung 5.1 mit Hilfe der Qubits $q1$ bis $q3$ dargestellt, auf die Server verteilt. Dabei erhält jeder Server genau eines der verschränkten Qubits. Hat schließlich jeder Server ein Qubit empfangen, ist die Initialisierungsphase beendet.

Algorithmus 5.1 Wahl eines Proposers, orientiert an [DP04].

```
1: quBit ← empfangenes Qubit aus |W_n⟩
2: proposer ← false
3: messErgebnis ← Messe(quBit)
4: if messErgebnis == 1 then
5:     proposer ← true
6: end if
```

5.1.3 Wahlphase des erweiterten Paxos

Vor Beginn des originalen Konsensusprotokolls wird eine Wahlphase eingefügt, die jeder teilnehmende Server nach Empfang eines Qubits durchläuft, siehe Algorithmus 5.1. Zu Beginn ist keiner der Server in der Rolle des Proposers, wie in Zeile 2 dargestellt. Zunächst wird der Zustands des empfangenen Qubits, wie in Abschnitt 2.1.4 beschrieben, in der Standardbasis gemessen und zwischengespeichert (Zeile 3). Es wird schließlich geprüft, ob das Messergebnis dem Wert 1 entspricht (Zeile 4). Ist dies der Fall, wird der Server zum Proposer ernannt, wie in Zeile 5 dargestellt. Andernfalls ändert sich seine Rolle nicht, und er bleibt passiv in seinem Handeln.

Da durch das Konstrukt des W-Zustands nur einer der n Server den Wert 1 messen kann, ist nach der initialen Wahlphase bereits entschieden, wer Proposer ist.

5.1.4 Ablauf des erweiterten Paxos

Jetzt kann der in Abschnitt 2.2.1 geschilderte Ablauf von Paxos beginnen, indem der ernannte Proposer mit dem Beginn einer Runde startet, eine *Prepare*-Nachricht generiert und diese an die anderen Server übermittelt [Lam01]. Hierbei muss, um den Erfolg der Runde nicht vorzeitig zu beenden, die Rundennummer größer gewählt werden als die bisherigen Rundennummern, sodass die Runde von den anderen Servern akzeptiert wird. Erst durch die Überbringung der *Prepare*-Nachricht an die anderen Server wird bekanntgegeben, wer die Wahl gewonnen hat. Zuvor hat ein Server nur darüber Kenntnis, welche Rolle er selbst für das Protokoll einnimmt. Sobald die Server wissen, wer der aktuelle Proposer ist, können sie auf erhaltene Anfragen von Clients mit der Adresse des Proposers antworten, sodass diese ihre Anfrage direkt an den Proposer stellen können.

Ist die Rundennummer des Proposers zu klein gewählt, oder fällt er aus, muss auf das Ende des aktuellen Zeitintervalls der zentralen Einheit gewartet werden. Daraufhin erzeugt die zentrale Einheit einen neuen W-Zustand mit n Qubits und verteilt diese auf die n Server, siehe beispielhaft Abbildung 5.1. Bei Erhalt der Qubits unterbrechen die Server den momentan laufenden Paxosalgorithmus und führen eine Neuwahl nach dem Algorithmus 5.1 in Abschnitt 5.1.3 durch. So besteht eine Wahrscheinlichkeit von $\frac{n-1}{n}$, dass ein anderer Server zum Proposer gewählt wird. Der originale Paxosalgorithmus sorgt dafür, dass die vorgeschlagene Eingabe der unterbrochenen Runde in die neue Runde mittels *Promise*-Nachricht überliefert wird [Lam01].

5.1.5 Vorteile bei Verwendung des W-Zustands

Durch die Verwendung des W-Zustands und die dadurch entstehende Verschränkung zwischen den teilnehmenden Servern ist es möglich, einen Proposer zu wählen ohne zusätzliche Kommunikation zwischen den Servern. So muss nicht, wie bei einer herkömmlichen Wahl eines klassischen Systems, untereinander kommuniziert und Konsens gefunden werden, um ein Wahlergebnis zu erhalten.

Ist der Algorithmus bereits mit der ersten Runde beendet, werden während des laufenden Protokolls $6n$ Nachrichten benötigt, und er ist somit gleichauf mit dem originalen Algorithmus von Paxos aus Abschnitt 2.2.1. Die erste Verteilung der Qubits des W-Zustands wird bereits in der Initialisierungsphase vorgenommen. Werden durch Neuwahlen zusätzliche Qubits im Nachhinein verteilt, müssen die jeweiligen Nachrichten der zentralen Einheit berücksichtigt werden.

Das durch den Zufall bestimmte Wählen eines Proposers kann für Paxos verwendet werden, da der Proposer bei Beginn einer Runde mit der *Prepare*-Nachricht und den darauffolgenden *Promise*-Nachrichten über bereits akzeptierte Eingaben erfährt. Dadurch wird, nach Lamports Bedingung, schließlich versucht die Eingabe zu entscheiden, die bereits in vorherigen, nicht abgeschlossenen Runden akzeptiert wurde [Lam01].

Des Weiteren ist durch die Eigenschaften des W-Zustands sichergestellt, dass höchstens ein Server zum Proposer gewählt wird. Auch wenn der gewählte Proposer ausfällt, wird durch den unveränderten Paxosalgorithmus die Konsistenz des Protokolls aufrechterhalten. Da durch den Ablauf von Zeitintervallen Neuwahlen ausgelöst werden, kann der Fortschritt des Protokolls ermöglicht und ein Verhungern des Systems verhindert werden. Paxos ist nicht darauf beschränkt sich ausschließlich auf den Wert eines einzelnen Bits zu einigen, wie es in den Algorithmen von D'Hondt und Panangaden [DP04] und Ben-Or und Hassidim [BH05] vorausgesetzt wird [ST17]. Zudem wird kein Zufall bei der Entscheidung der Eingabe selbst verwendet.

Der Wahlmechanismus kann auch für Multi-Paxos, aus Abschnitt 2.2.2, eingesetzt werden. So besteht die Möglichkeit, dass ein erwählter Proposer innerhalb eines gewählten Zeitintervalls mehrere Eingaben vorschlagen und entscheiden lassen kann.

5.1.6 Mögliche Probleme

Die Größe des Zeitintervalls für das Auslösen von Neuwahlen muss mit Bedacht gewählt werden. Ist das Intervall zu groß und der Proposer fällt bereits zu Beginn aus, wird nur ein sehr langsamer Fortschritt des Protokolls erreicht. Ist das Intervall hingegen zu klein, wird der jeweils aktuelle Proposer möglicherweise während seiner Runde regelmäßig unterbrochen und kein Fortschritt wird erreicht. Eine Alternative ist, die zentrale Einheit an allen Nachrichten des Konsensusprotokolls teilhaben zu lassen. Die zentrale Einheit lässt nach Empfang einer jeden Nachricht einen neuen Timeout starten. Läuft der Timeout ab, nimmt sie an, dass der Proposer ausgefallen ist. Daraufhin erzeugt sie einen neuen W-Zustand und verteilt dessen Qubits. Auch hier unterbrechen die Server bei Erhalt eines Qubits den Paxosalgorithmus und führen den Wahlalgorithmus 5.1 aus. Hierfür werden neben den Quantenkanälen noch n weitere klassische Kanäle von jedem Server zur zentralen Einheit benötigt. Bei jeder Kommunikation zwischen den Servern muss eine zusätzliche Nachricht für die zentrale Einheit erzeugt werden. Des Weiteren gestaltet sich die Wahl eines passenden Timeouts in einem asynchronen System ähnlich schwierig wie die Wahl eines Zeitintervalls. Jedoch wäre eine Neuwahl mit dieser Alternative weniger willkürlich.

In der Theorie kann der Fall auftreten, dass unendlich oft ein ausgefallener Server zum Proposer ernannt wird, sodass sich kein Fortschritt ereignet. So kann auch die Anzahl der Nachrichten unbegrenzt zunehmen. Wie in Abschnitt 2.2.1 beschrieben, könnten, ohne die Fortschrittlichkeit von Paxos zu stoppen, maximal $\frac{n-1}{2}$ Server ausfallen [MJ13]. Würde dies der Fall, sein und die Anzahl der ausgefallenen Server ändert sich nicht, wäre die Wahrscheinlichkeit bei jeder Wahl einen der ausgefallenen Server zu wählen bei $\frac{n-1}{2} \cdot \frac{1}{n} = \frac{n-1}{2n}$. Wie in Abschnitt 2.1.6 erläutert, ist $\frac{1}{n}$ die Wahrscheinlichkeit einen der n Zustände des W-Zustands zu messen.

5.1.7 Betrachtung byzantinischer Fehler

Der Paxos Algorithmus selbst ist nicht für byzantinische Fehler ausgelegt [Lam01]. Auch die Erweiterung ist nicht geschützt vor solch einer Art von Ausfällen. Als Beispiel könnte ein bösartiger Server, trotzdem er nicht gewählt wurde, nach jeder Wahlperiode eine höhere Rundennummer wählen als der aktuelle Proposer und selbst eine Runde beginnen [CV17]. So könnte er entweder Eingaben vorschlagen, die nicht denen der Clients entsprechen oder keine weiteren Aktionen durchführen und so das System am Fortschreiten hindern. Eine Möglichkeit dieses Vorgehen einzudämmen, ist, die Server darauf zu beschränken, nur die erste *Prepare*-Nachricht nach der Wahlperiode anzunehmen und weitere zu ignorieren. Allerdings könnte zum einen der byzantinische Server schneller als der ernannte Proposer sein, zum anderen könnte der byzantinische Server selbst zum Proposer gewählt werden [CV17]. So müsste der byzantinische Server nach wenigen Zeitintervallen erkannt und in den darauf folgenden Runden von der Gruppe sowie der zentralen Einheit ignoriert werden.

Ein bekanntes Konsensusprotokoll, welches Paxos ähnelt und einer begrenzten Anzahl byzantinischer Fehler standhält, wurde von Castro, Liskov et al. entwickelt [CL+99]. Sie behelfen sich der Kryptographie und verwenden unter anderem digitale Signaturen und Hashing zur Verifizierung von Nachrichten. Des Weiteren ist die benötigte Anzahl der funktionierenden Server im Verhältnis zu den fehlerhaften größer. Für dieses Protokoll gilt mit insgesamt n Servern und davon f ausgefallenen, beziehungsweise byzantinischen, $n > 3f$. Der Aufwand für die Absicherung gegen byzantinische Server ist im Vergleich zu simpel ausfallenden Servern somit sehr groß.

5.2 Paxos: Rundennummervergabe mit Superposition

Die zweite Variante von Paxos mit Quantencomputing ist in den ursprünglichen Algorithmus integriert. Hierbei wird die Idee von Chlebus et al. eingesetzt und die quantenmechanischen Eigenschaften von Superposition und Verschränkung verwendet [CKS10]. Jeder Server kann, wie im ursprünglichen Paxosprotokoll [Lam01], anders als in der vorherigen Erweiterung 5.1, zu jeder Zeit eine Eingabe vorschlagen.

5.2.1 Umgebung und zusätzliche Mechanismen

Es wird ebenfalls von einer asynchronen Umgebung ausgegangen [Lam01; VA15]. Die Server fallen durch Halten aus. Dies unterscheidet sich von der angenommenen Umgebung des Algorithmus von Chlebus et al. [CKS10].

Algorithmus 5.2 Vereinfachte Darstellung der Überprüfung einer empfangenen *Prepare*($|r\rangle$)-Nachricht, orientiert an [CKS10; Lam01].

1: $|c\rangle \leftarrow |0\rangle$
2: Conditional-Not($|r\rangle > |m\rangle, |c\rangle$)
3: $c \leftarrow$ Messe($|c\rangle$)
4: **if** $c == 1$ **then**
5: Entangled-Copy($|rmv\rangle$, *Promise*)
6: sende *Promise*($|rmv\rangle$) an Proposer der Runde $|r\rangle$
7: **end if**

Da ein Großteil der übermittelten Nachrichten Quantenzustände enthalten, werden zwischen allen Servern zusätzlich Quantenkanäle benötigt. So werden, neben den bereits vorhandenen klassischen Kanälen, $n(n-1)$ bidirektionale Quantenkanäle verwendet. Diese Quantenkanäle sind jedoch weniger zuverlässig als die in der ersten Erweiterung in Abschnitt 5.1. Nachrichten können verloren gehen oder vervielfacht werden, jedoch nicht verändert, wie es auch bei Paxos gegeben ist [Lam01].

Des Weiteren wird für die Vervielfachung der Rundennummer und anderer Quantenregister der Entangled-Copy-Operator [CKS10] von Chlebus et al. zur Erstellung aller Nachrichten verwendet, siehe Abschnitt 2.2.3. Als Rundennummer verwendet ein Server in dieser Erweiterung die gleichverteilte Superposition aller Zahlen von 1 bis n^3 in der Standardbasis [BH05; CKS10]:

$$|r\rangle = \frac{1}{2^{\frac{3}{2}\log_2 n}} \sum_{a=1}^{n^3} |a\rangle \tag{5.2}$$

$$= \frac{1}{\sqrt{2^{3\log_2 n}}} \sum_{a=1}^{n^3} |a\rangle \tag{5.3}$$

$$= \frac{1}{\sqrt{n^3}} \sum_{a=1}^{n^3} |a\rangle \tag{5.4}$$

Wobei n auch hier der Anzahl der Server entspricht. Mit Hilfe von $\log_2 n$ wird die für die Dezimalzahl n benötigte Anzahl an Bits berechnet. Multipliziert mit 3, wie in Gleichung (5.4) dargestellt, ergibt die Anzahl Bits für die Darstellung von n^3. Mit 2 als Basis werden schließlich, wie innerhalb des Grundlagenabschnittes 2.1 erläutert, alle möglichen binären Zustände beschrieben.

5.2.2 Ablauf des erweiterten Paxos

Um eine Runde zu beginnen und so den anderen Servern eine Eingabe vorschlagen zu können, muss ein Server zunächst eine Rundennummer r wählen, wie in Lamports Paxos [Lam01]. Hierbei verwendet ein Server als Rundennummer die gleichverteilte Superposition aller Zahlen von 1 bis n^3, wie in Abschnitt 5.2.1 erläutert [BH05; CKS10]. Die generierte Rundennummer $|r\rangle$ wird mit einer *Prepare*-Nachricht an alle Server über die Quantenkanäle übermittelt, wie im Kommunikationsverlauf in Abbildung 5.2a zu sehen. Wie bei Lamports Paxos [Lam01], Abschnitt 2.2.1, überprüft ein Acceptor bei Erhalt der *Prepare*-Nachricht die Aktualität der

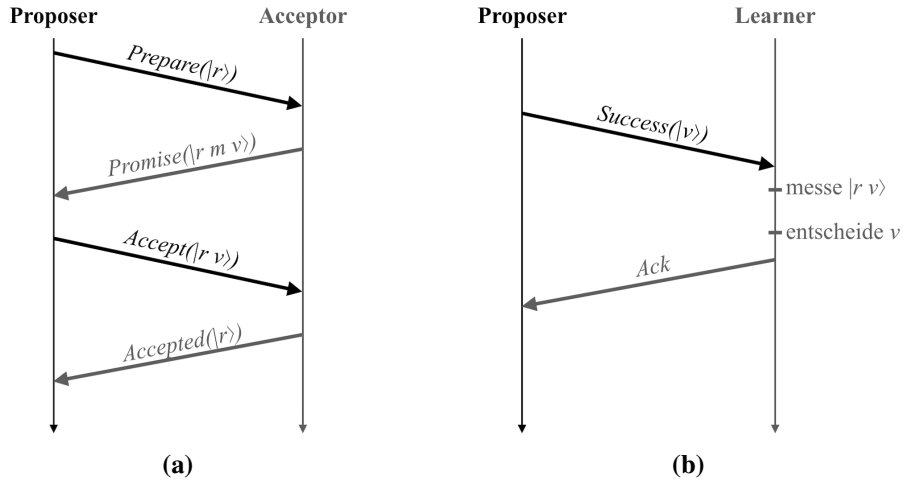

Abbildung 5.2: Kommunikationsverlauf von Paxos mit Quantencomputing, orientiert an [DLL00]. (a) Nachrichtenaustausch zwischen Proposer und Acceptor zur Entscheidung einer Eingabe. (b) Bekanntgabe der entschiedenen Eingabe an Learner sowie dessen Bestätigung.

Rundennummer, siehe Algorithmus 5.2. Hierfür wird zunächst ein einzelnes Qubit $|c\rangle$ auf $|0\rangle$ gesetzt, siehe Schritt 1. Daraufhin wird in Schritt 2 der Operator Conditional-Not($|r\rangle > |m\rangle, |c\rangle$) [CKS10] von Chlebus et al. genutzt. Ist $|r\rangle$ höher als die Rundennummer $|m\rangle$ der zuletzt akzeptierten Runde, wird $|c\rangle$ negiert. Schließlich wird $|c\rangle$ in Schritt 3 gemessen. Ist $c = 1$, wird die neue Rundennummer $|r\rangle$, die ältere Rundennummer $|m\rangle$ und die in der älteren Runde akzeptierte Eingabe $|v\rangle$, falls vorhanden, nach Chlebus et al. verschränkt, in eine *Promise*-Nachricht kopiert (Schritt 5) und dem Proposer übermittelt (Schritt 6) [CKS10]. Analog zu Lamports Paxos [Lam01], siehe Abschnitt 2.2.1.

Nachdem der Proposer mehrheitlich *Promise*-Nachrichten empfangen hat, nimmt er, falls erhalten, die neuste Eingabe $|v\rangle$, wie in Abbildung 5.2a zu sehen, und erzeugt durch mehrmaliges, verschränktes Kopieren *Accept*($|rv\rangle$)-Nachrichten. Diese sendet er, wie in dem originalen Paxos, an die Acceptors [Lam01]. Erneut überprüfen diese die Aktualität der Runde, analog zu Algorithmus 5.2, und schicken bei Gültigkeit ein *Accepted*($|r\rangle$) zurück.

Wenn auch hier der Proposer eine Mehrheit an *Accepted*-Nachrichten empfangen hat, gilt die Eingabe $|v\rangle$ als entschieden, wie in Lamports Protokoll [Lam01]. Allerdings wird die tatsächliche Entscheidung, nach der Messung, erst zu einem späteren Zeitpunkt für den Proposer durchgeführt. Zunächst sendet er *Success*($|v\rangle$) via Entangled-Copy an alle Server und wartet auf deren Bestätigung (*Ack*), wie in Abbildung 5.2b dargestellt. Bei Empfang der *Success*-Nachricht werden die Server zu Learnern, messen $|r\rangle$ und $|v\rangle$ in der Standardbasis und entscheiden schließlich v. Dabei ist jeder der n^3 verschiedenen Zustände der Rundennummer $|r\rangle$ mit Wahrscheinlichkeit $\frac{1}{n^3}$ das Messresultat. Erhält der Proposer von allen das *Ack*, kann auch er die Rundennummer sowie die Eingabe messen. Durch die Verschränkung zwischen den Servern erhalten alle Teilnehmer dieselbe Rundennummer r und Eingabe v als Messergebnis.

5.2.3 Vorteile bei Verwendung der Superposition

Durch die Superposition aller Zahlen von 1 bis n^3 in der Standardbasis als Rundennummer jedes Proposers, ist der Zahlenbereich sehr groß. So ist es eher gering wahrscheinlich, dass dieselbe Rundennummer von mehr als einem Proposer gemessen wird. Ist dies dennoch der Fall, kann die Identifikationsnummer des Servers miteinbezogen werden. Des Weiteren ist es möglich, dass durch das zufällige Wählen einer Rundennummer, im Gegensatz zu einem selbstständigen Wählen einer noch höheren Rundennummer, ein Proposer seltener von einem anderen Server übertrumpft und unterbrochen wird.

Zudem wird auch in dieser Erweiterung über eine Eingabe aus einer Clientanfrage entschieden und nicht über einen zufälligen Wert. Die Eingabe kann ursprünglich aus klassischen Informationen bestehen. Für den Transport über Quantenkanäle müssen die dafür benötigten klassischen Bits, wie in Abschnitt 2.1.7 beschrieben, zunächst in Qubits umgewandelt werden [GIN18]. Die Umwandlung zurück in klassische Information geschieht erst durch die Messung bei der Entscheidung von $|v\rangle$. Dadurch können Umwandlungsschritte reduziert werden. $|v\rangle$ kann so ein einzelnes Qubit oder ein Quantenregister, wie $|r\rangle$ und $|m\rangle$, mit mehreren Qubits sein. Die Eingabe kann ebenso von vornherein ein Quantenzustand sein.

Die Anzahl der Nachrichten ist gleich zu der des klassischen Paxos [DLL00; Lam01], dargestellt in Abschnitt 2.2.1. So werden bei einem fehlerfreien Durchlauf $6n$ Nachrichten benötigt, diese werden jedoch über Quantenkanäle anstatt klassischer Kanäle übermittelt. Für Fehlerfälle ist die Zahl der Nachrichten nach oben unbegrenzt. Ebenfalls müssen hier über die Hälfte der teilnehmenden Server funktionieren und dürfen nicht ausfallen, sodass Mehrheiten gebildet werden können [MJ13].

Eine weitere Besonderheit, verglichen zu dem ersten Zeitintervall der ersten Erweiterung, Abschnitt 5.1, wird die Verschränkung, wie in dem Protokoll von Chlebus et al. [CKS10], während des Algorithmus erzeugt und benötigt keinerlei Vorbereitung in der Initialisierungsphase. Weiterhin bedarf das erweiterte Protokoll keiner zentralen Einheit mit zuverlässigen Quantenkanälen. So findet der gesamte Ablauf bei und zwischen den Servern der Gruppe selbst statt.

Wie bei Chlebus et al., Abschnitt 2.2.3, kann von einem Widersacher ausgegangen werden, der die Kommunikation mitlesen kann und danach bestimmt, welche Server während der Laufzeit ausfallen, um den Ablauf zu beeinträchtigen [CKS10]. Da sich die Rundennummern auch hier während der Kommunikation in der Superposition befinden, lassen sich für den Widersacher während des Lesens keine Rückschlüsse darüber ziehen, welcher Proposer nach der Messung schließlich die höchste Rundennummer besitzt.

Da die Nachrichtentypen äquivalent zu den Nachrichtentypen von Paxos [DLL00; Lam01], siehe Abschnitt 2.2.1, sind, kann auch diese Erweiterung analog für Multi-Paxos aus Abschnitt 2.2.2 verwendet werden.

5.2.4 Mögliche Probleme

Misst, wie in der Erweiterung angenommen, der Proposer die entschiedene Eingabe erst, nachdem er von allen anderen Servern ein *Ack* empfangen hat, kann die Clientanfrage erst zu einem späteren Zeitpunkt als bisher bei klassischem Paxos [DLL00; Lam01] angenommen beantwortet werden. Eine Möglichkeit ist, dass nicht nur der Proposer, sondern auch die Acceptors dem Client antworten,

um weitere Verzögerungen zu minimieren. Außerdem wird in der Theorie angenommen, dass der Proposer die *Success*-Nachrichten wiederholt versendet, bis er von allen Servern eine Antwort erhält [DLL00]. Bei Ausfall eines Servers würde der Client so nie eine Antwort erhalten.

Eine weitere Möglichkeit besteht darin, dass der Proposer die Eingabe und die zugehörige Rundennummer bereits nach Erhalt der Mehrheit an *Accepted*-Nachrichten misst. Da die *Success*-Nachricht nur die Eingabe selbst enthält, siehe Abbildung 5.2b, kann diese vor dem Transport mittels Quantenkanälen erneut in einen Quantenzustand umgewandelt werden, siehe Abschnitt 2.1.7. Vorausgesetzt die Eingabe entsprach ursprünglich bereits klassischen Informationen, sodass eine Wiederherstellung des erstmalig erzeugten Quantenzustands ohne Informationsverlust möglich ist.

Die Frage, wie zwei gleichverteilte Superpositionen miteinander verglichen werden, wie es für den Conditional-Not($|r\rangle > |m\rangle, |c\rangle$)-Operator [CKS10] von Chlebus et al. für den Vergleich von Rundennummern verwendet wird, bleibt im Rahmen dieser Arbeit offen. Angenommen, ein Quantenregister mit n Qubits befindet sich in Superposition, die Amplituden der Zustände sind jedoch nicht gleichverteilt, sondern haben unterschiedliche Werte, sodass ein Zustand nach Messung wahrscheinlicher ist als die anderen. Wird dieses mit einem anderen Quantenregister der gleichen Größe, in Superposition und mit unterschiedlichen Amplituden verglichen, kann wohl angenommen werden, dass eines der beiden Quantenregister mit einer bestimmten Wahrscheinlichkeit größer ist, beziehungsweise ein größeres Messergebnis ergibt als das andere. Durch die Bedingung der gleichverteilten Superposition aller Zahlen von 1 bis n^3 in der Standardbasis haben jedoch alle Amplituden denselben Wert, keiner der Zustände der beiden Quantenregister ist wahrscheinlicher. So liegt die Vermutung nahe, dass es sich nur um Zufall handeln kann, welche der verglichenen Rundennummern als größere ausgewertet wird. Wichtig ist insbesondere, dass bei allen Servern das gleiche Resultat erzeugt wird, um einen Konsens zu ermöglichen. Würde hingegen $|r\rangle > |m\rangle$ stets *false* ausgeben, sodass $|r\rangle$ immer kleiner oder gleich $|m\rangle$ ist, würden in keinem Fall *Promise*-Nachrichten versendet werden, siehe Algorithmus 5.2, und somit kein Fortschritt möglich sein. Dies kann jedoch nicht im Sinne von Chlebus et al. sein, da auch ihr Protokoll sonst keinen Konsens finden würde [CKS10]. Auf Nachfrage bei einem der Autoren konnte leider keine aufschlussreiche Antwort gegeben werden.

5.2.5 Betrachtung byzantinischer Fehler

Wie bereits in Abschnitt 5.2.3 erläutert, kann ein Widersacher keine Informationen durch Lesen der Nachrichten erhalten, um auf das resultierende Messergebnis schließen zu können [CKS10]. Allerdings schützt dies nicht vor byzantinischen Servern innerhalb der Gruppe. So kann auch hier ein byzantinischer Server Proposer der aktuellen Runde werden, indem er als Rundennummer n^3 wählt und ein Voranschreiten des Algorithmus verhindert, wie bereits in der ersten Erweiterung in Abschnitt 5.1 erwähnt [CV17]. Eine weitere Möglichkeit ist, dass er Eingaben vorschlägt, die in keiner der Clientanfragen enthalten waren.

So müsste auch hier Kryptographie, beispielsweise mit digitalen Signaturen und Hashing, verwendet werden, wie bei dem Protokoll von Castro, Liskov et al. [CL+99]. Zudem wird wohl auch hier für das Tolerieren von byzantinischen Fehlern eine höhere Mindestanzahl an funktionierenden Servern als bisher benötigt.

5.3 Zusammenfassende Erkenntnisse

Eine geringere Laufzeit, gemessen an der Anzahl der übermittelten Nachrichten, kann durch die beiden Erweiterungen mit Quantencomputing in den Abschnitten 5.1 und 5.2 nicht festgestellt werden. Durch die asynchrone Umgebung, den möglichen Verlust an Nachrichten und der Bedingung, eine gewisse Anzahl an Ausfällen tolerieren zu können, werden die verschiedenen Nachrichtentypen des klassischen Paxos [DLL00; Lam01] in den Erweiterungen weiterhin benötigt.

Dabei können die verschiedenen Nachrichtentypen in drei Teile gegliedert werden. Im ersten Teil werden durch *Prepare*- und *Promise*-Nachrichten eine neue Runde begonnen und Informationen über vorherige Runden gesammelt, siehe Abbildung 2.1 [DLL00; Lam01]. Der zweite Teil dient dazu, durch *Accept* und *Accepted* eine Eingabe vorzuschlagen und Stimmen für diese einzuholen. Der dritte Teil dient mit seinen *Success*- und *Ack*-Nachrichten dazu, die Entscheidung zu verbreiten [DLL00].

Die erste Erweiterung aus Abschnitt 5.1 verbessert insbesondere die Fortschrittlichkeit in gewissem Maße, indem durch regelmäßige Wahlphasen nur ein einzelner Proposer zur gleichen Zeit existiert. Die Problematik des häufig gegenseitigen Unterbrechens, siehe Abschnitt 2.2.1, kann so umgangen werden [Lam96]. Die zweite Erweiterung aus Abschnitt 5.2 lässt die Proposer durch Zufall eine Rundennummer wählen. So könnte auch hier ein gegenseitiges Unterbrechen reduziert werden. Des Weiteren kann ein Widersacher bei Lesen der Nachrichten über Quantenkanäle keine Rückschlüsse daraus ziehen, welche die aktuelle Runde ist, da sich die Rundennummern während der Kommunikation in Superposition befinden [CKS10]. Damit kann von einem stärkeren Fehlermodell ausgegangen werden.

Allerdings muss beachtet werden, dass es sich bei den beiden Erweiterungen um rein theoretische Maßnahmen handelt. Die tatsächliche Umsetzbarkeit dieser Erweiterungen ist somit fraglich. So ist auch Paxos [Lam01] selbst sehr abstrakt und theoretisch, mit viel Interpretationsfreiraum, gehalten. Die Umsetzungen weichen daher häufig voneinander ab [ACD16]. Des Weiteren kann die Stabilität von Quantenzuständen heutzutage nur für kurze Zeit aufrecht erhalten werden, siehe Abschnitt 2.1.11 [Hom18]. Ob die Lebensdauer der durch Qubits übermittelten Informationen in solch einem asynchronen, verteilten System daher in naher Zukunft ausreicht, ist fraglich. Zudem wird für die Wiederherstellung eines Servers nach einem Absturz persistenter Speicher benötigt [CGR07]. Während einer Runde muss sich ein Server unter anderem entschiedene und akzeptierte Eingaben sowie deren zugehörigen Rundennummern persistent speichern [DLL00]. Eine Möglichkeit, solche verschränkten Quantenzustände für längere Zeit zu speichern, gibt es momentan jedoch nicht [RP11].

6 Verwendung von Quantencomputing in Commit-Protokollen

In diesem Kapitel werden Erweiterungen für das 2-Phasen-Commit-Protokoll [Gra78] und dessen kooperatives Terminierungsprotokoll, aus den Abschnitten 2.3.1 und 2.3.2, sowie für das 3-Phasen-Commit-Protokoll [Ske81] und dessen Terminierungsprotokoll, aus den Abschnitten 2.3.3 und 2.3.4, präsentiert. Hierbei enthalten die vorgestellten Erweiterungen Konzepte des Quantencomputings. Es soll zum einen untersucht werden, ob der Fall der ungelösten Blockade des 2-Phasen-Commit-Protokolls in Abschnitt 2.3.1 mit Hilfe der Quantenmechanik aufgelöst werden kann. Zum anderen soll, ebenso wie bei der Erweiterung von Paxos [Lam98] in Abschnitt 5.1, anhand der Erweiterungen des 3-Phasen-Commit-Protokolls und des Terminierungsprotokolls untersucht werden, ob eine Reduzierung des Kommunikationsaufwands möglich ist.

In dem folgenden Abschnitt 6.1 werden zunächst zwei verschiedene Erweiterungen in Bezug auf das 2-Phasen-Commit-Protokoll und dessen kooperativen Terminierungsprotokoll vorgestellt. Dabei werden mögliche Probleme diskutiert. Anschließend wird in Abschnitt 6.2 die Erweiterung des 3-Phasen-Commit-Protokolls und dessen zugehöriges Terminierungsprotokoll, siehe Abschnitt 6.2, präsentiert. Darauffolgend werden Vorteile, mögliche Problematiken und eine zusätzliche Erweiterung vorgestellt. Zuletzt werden die gewonnenen Erkenntnisse aus der Entwicklung der Erweiterungen für die beiden Commit-Protokolle präsentiert.

6.1 2-Phasen-Commit-Protokoll: Deblockieren mit GHZ-Zustand

Im Folgenden werden zwei Varianten des 2-Phasen-Commit-Protokolls aus Abschnitt 2.3.1 mit Quantencomputing präsentiert. In der ersten Erweiterung in Abschnitt 6.1.3 wird untersucht, ob es möglich ist, mit Hilfe von Verschränkung eine entstehende Blockade zu lösen, ohne das kooperative Terminierungsprotokoll aus Abschnitt 2.3.2 anwenden zu müssen. In der zweiten Erweiterung in Abschnitt 6.1.4 wird ebenfalls mit der Verwendung von Verschränkung untersucht, ob der blockierende Fall, Antwortmöglichkeit (c) in Abschnitt 2.3.2, des kooperativen Terminierungsprotokolls aufgelöst werden kann. Für beide Erweiterungen wird der GHZ-Zustand verwendet, der in Abschnitt 2.1.6 präsentiert wird. Dieser verschränkte Zustand wird dazu verwendet, einen einheitlichen Konsens in Bezug auf die Entscheidung, ob Commit oder Abort einer Transaktion, finden zu können, obwohl der Koordinator nicht erreichbar ist. Mögliche Probleme werden in Abschnitt 6.1.5 aufgeführt.

6.1.1 Umgebung und zusätzliche Mechanismen

In beiden Erweiterungen wird, wie im originalen 2-Phasen-Commit-Protokoll aus Abschnitt 2.3.1, eine teilweise synchrone Umgebung angenommen [BHG87; ST17]. So können auch Kommunikationsfehler auftreten. Die Prozesse fallen durch Halten aus [BHG87; Zha07].

Da der Koordinator neben herkömmlichen Nachrichten auch Qubits übermittelt, werden n Quantenkanäle benötigt, sodass zwischen dem Koordinator und den Teilnehmern eine 1:n Sternverbindung besteht, siehe beispielhaft die grauen Quantenkanäle mit den Qubits $q1$ bis $q3$ in den Abbildungen 6.1 und 6.2. Zudem wird in den Abbildungen 6.1 und 6.2 mit Hilfe von schwarzen Kanälen die klassische Kommunikation zwischen den Teilnehmern und dem Koordinator dargestellt.

6.1.2 Initialisierungsphase beider Erweiterungen

Der Ablauf der Generierung und Verteilung des GHZ-Zustands ist bei beiden Erweiterungen gleich und ähnelt der Idee von D'Hondt und Panangaden [DP04]. Der Koordinator erzeugt zur Initialisierungsphase des 2-Phasen-Commit-Protokolls einen GHZ-Zustand, besteht aus n Qubits. Wobei n der Anzahl aller Teilnehmer, ohne Koordinator, entspricht:

$$|GHZ_n\rangle = \frac{1}{\sqrt{2}}(|\underbrace{0...0}_{n \text{ Qubits}}\rangle + |1...1\rangle) \qquad ,n : \#\text{ Teilnehmer} \tag{6.1}$$

Verteilt wird der GHZ-Zustand schließlich mittels *Prepare*-Nachrichten. Mit diesen Nachrichten übergibt der Koordinator zusätzlich jedem Teilnehmer jeweils ein Qubit des GHZ-Zustands, wie in den Abbildungen 6.1 und 6.2 mit den Qubits $q1$ bis $q3$ für die Teilnehmer T1 bis T3 dargestellt. Der Zeitpunkt der Messung der Qubits wird in den beiden Erweiterungen unterschiedlich gehandhabt. Gemessen wird in der Standardbasis, sodass das Ergebnis 0 äquivalent zu einem Abort ist, Gleiches gilt für das Ergebnis 1, welches äquivalent für ein Commit steht. Mit jeweils fünfzigprozentiger Wahrscheinlichkeit erhalten alle bei der Messung entweder ein Commit oder ein Abort.

6.1.3 GHZ-Zustand statt Terminierungsprotokoll

Läuft der Timeout eines Teilnehmers nach Senden des *Yes*, siehe den Kommunikationsverlauf in Abbildung 2.2, ab, nimmt er an, dass der Koordinator nicht erreichbar ist. Ohne Erweiterung und ohne kooperatives Terminierungsprotokoll müsste er nun warten, bis der Koordinator ihm wieder antwortet [BHG87]. Um nun eine Blockade zu vermeiden bis der Koordinator mit einer Entscheidung antwortet, misst der Teilnehmer das erhaltene Qubit und entscheidet die Transaktion entsprechend mit einem Commit oder Abort. Die Entscheidung übermittelt der Teilnehmer an den Koordinator, der die Nachricht erhält, sobald er wieder erreichbar ist.

Diese Art der Deblockierung ist nur möglich, wenn der Koordinator abstürzt, während er sich noch in der Abstimmungsphase befindet und alle Teilnehmer mit einem *Yes* abgestimmt haben. In der Abstimmungsphase hat der Koordinator selbst noch keine Entscheidung getroffen, beziehungsweise konnte das Abstimmungsergebnis noch nicht persistent speichern, sodass die Konsistenz in diesem Fall nicht verletzt ist. Die Teilnehmer stimmen alle einem Commit zu, erfahren einen Timeout und messen schließlich alle das selbe Ergebnis. Ist der Koordinator wieder hergestellt, sendet er, wie in Abschnitt 2.3.1 erläutert, erneut *Prepare*-Nachrichten an alle [ST17]. Die Teilnehmer können ihm wiederum mit dem Ergebnis der Entscheidung antworten, sodass auch der Koordinator entsprechend die Transaktion entscheidet. Für diesen Fall finden die Teilnehmer einen Konsens, ohne zusätzliche Kommunikation zu benötigen.

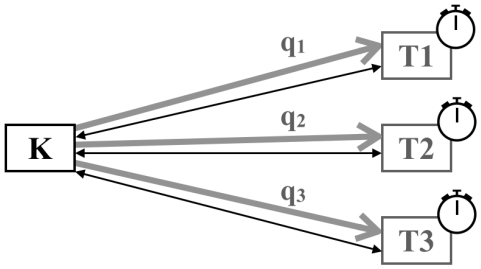

Abbildung 6.1: Kommunikationskanäle der ersten Erweiterung des 2-Phasen-Commit-Protokolls. Dünne, schwarze Pfeile: klassische Kommunikationskanäle. Dicke, graue Pfeile: Quantenkanäle.

Ist die Entscheidung des Koordinators bereits vor seinem Absturz gefallen, könnte der Teilnehmer durch Zufall ein gegensätzliches Ergebnis messen. Zudem könnte eine Inkonsistenz auftreten, wenn es sich nicht um einen Ausfall des Koordinators handelt, sondern um ein Verbindungsproblem. Dadurch könnte der Koordinator bereits in der Zwischenzeit eine Entscheidung gefällt haben, sodass der Teilnehmer mit Timeout auch hier wieder durch Zufall ein gegensätzliches Ergebnis messen könnte. Zu beachten ist auch der Fall, dass ein anderer Teilnehmer möglicherweise für ein *No* stimmt. Erfährt der Koordinator nach seiner Wiederherstellung davon, wenn er dies nicht bereits wusste, wird die Transaktion mit einem Abort entschieden, wie in Abschnitt 2.3.1 beschrieben [BHG87; ST17]. Da sich die Teilnehmer im 2-Phasen-Commit-Protokoll nicht untereinander austauschen, siehe Abbildung 6.1, misst auch hier ein Teilnehmer nach Timeout sein Qubit und kann per Zufall ein Commit entscheiden, welches ebenfalls die Konsistenz verletzt.

6.1.4 Terminierungsprotokoll mit GHZ-Zustand

In Abschnitt 2.3.2 wird aufgezeigt, dass, wenn alle Teilnehmer für ein Commit stimmen (*Yes*), jedoch daraufhin keine Antwort des Koordinators empfangen haben, sie untereinander, trotz kooperativem Terminierungsprotokoll, keine Entscheidung treffen können (Fall (c) der Antwortmöglichkeiten) [BHG87; BN09]. Um eine Blockade zu vermeiden, messen die Teilnehmer in diesem Szenario jeweils ihr erhaltenes Qubit. Entsprechend dem Messergebnis entscheiden sie ein Commit oder ein Abort der Transaktion. Durch die Eigenschaften des GHZ-Zustands erhalten schließlich alle dasselbe Messergebnis und haben einen Konsens gefunden. Ist der Koordinator wiederhergestellt, überliefern die Teilnehmer ihm ihre Entscheidung, sodass dieser entsprechend entscheiden kann.

Es gilt ebenso hier, dass der GHZ-Zustand nur verwendet werden kann, wenn alle Teilnehmer für ein Commit gestimmt haben, sich gegenseitig erreichen können, wie in Abbildung 6.2 mit den schwarzen, klassischen Kanälen zwischen den Teilnehmern T1 bis T3 dargestellt, und der Koordinator noch keine Entscheidung getroffen hat. Stürzt der Koordinator in der Abstimmungsphase ab, und eine Entscheidung konnte noch nicht persistent gespeichert werden, antworten ihm die Teilnehmer, nach erneutem Senden der *Prepare*-Nachrichten, mit ihrer Entscheidung. Dadurch ist es ihm möglich, dieselbe Entscheidung zu treffen. In diesem Fall wird keine zusätzliche Kommunikation zwischen den Teilnehmern benötigt, um sich bei der Entscheidung abzustimmen.

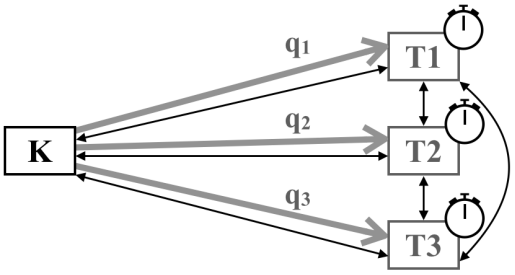

Abbildung 6.2: Kommunikationskanäle der zweiten Erweiterung des 2-Phasen-Commit-Protokolls. Dünne, schwarze Pfeile: klassische Kommunikationskanäle. Dicke, graue Pfeile: Quantenkanäle.

Stürzt der Koordinator jedoch ab, nachdem er bereits eine Entscheidung getroffen hat, diese jedoch nicht mehr verbreiten konnte, könnte bei der Messung des GHZ-Zustands das Gegenteil entschieden werden. Inkonsistenz ist so mit einer Wahrscheinlichkeit von 50% möglich. Ist ein Teilnehmer während des Austauschs für niemanden erreichbar, da er nicht auf *Decision-Request*-Nachrichten antwortet, könnte es sein, dass dieser mit *No* abstimmt. Ein Abort steht in diesem Fall fest, dieses kann jedoch erst bekannt gegeben werden, sobald der Teilnehmer wieder erreichbar ist. Eine Entscheidung anhand des GHZ-Zustands wäre somit unzulässig. Denn auch hier könnte eine Messung zufällig zu einem Commit bei den anderen Teilnehmern führen.

6.1.5 Weitere mögliche Probleme

Für die blockierten Teilnehmer ist im Allgemeinen nicht ersichtlich, in welchem Zustand der Koordinator abgestürzt ist und so bereits eine Entscheidung getroffen hat. Ein zuverlässiges Selbstentscheiden, wann das erhaltene Qubit in den Erweiterungen, in den Abschnitten 6.1.3 und 6.1.4, gemessen und verwendet werden soll und wann nicht, ist somit nicht möglich.

Des Weiteren kommt hinzu, dass in beiden Erweiterungen angenommen wird, dass der Koordinator selbst stets für ein Commit stimmt, das heißt vor Senden der *Prepare*-Nachrichten somit schon bereit für den Commit einer Transaktion ist. Falls er so für ein Abort ist, kann er zu Beginn des 2-Phasen-Commit-Protokolls bereits zur Entscheidungsphase springen und *Abort*-Nachrichten an alle Teilnehmer senden. Allerdings wird im Allgemeinen davon ausgegangen, dass der Koordinator erst in der Abstimmungsphase seine Stimme abgibt und so auch mit einem *No* abstimmen kann [RK98; ST17]. Ein Konsens allein unter den Teilnehmern ist damit nicht ausreichend. So muss auch hier auf die Antwort des Koordinators gewartet werden, um sicher zu gehen, dass er sich nicht entgegen des Messergebnisses des GHZ-Zustands entscheidet. Ergibt das Messergebnis der Teilnehmer ein Commit, kann der Fall eintreten, dass der Koordinator sich zwar nach einer Wiederherstellung noch in der Abstimmungsphase befindet, jedoch selbst mit einem *No* abstimmt. Die Konsistenz kann auch hier verletzt werden. Dabei wird kein Vorteil erlangt, wenn der Koordinator selbst ebenfalls ein Qubit des GHZ-Zustands zugewiesen bekommt. Der GHZ-Zustand wird dafür auf $n + 1$ Qubits erweitert. Würde der Koordinator schließlich sein Qubit nach Wiederherstellung messen, und das Resultat wäre ein Commit, widerspricht es seiner Abstimmung, wenn er ein *No* abgibt.

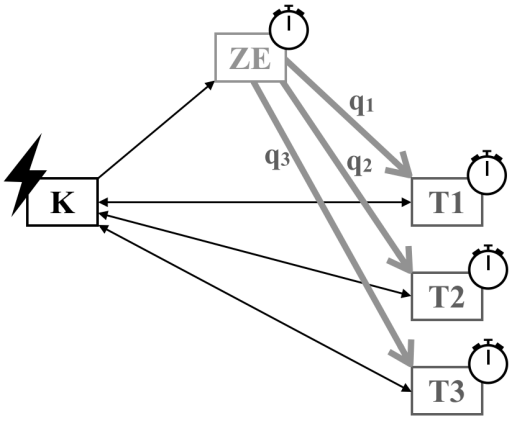

Abbildung 6.3: Kommunikationskanäle der Erweiterung des 3-Phasen-Commit-Protokolls. Dünne, schwarze Pfeile: klassische Kommunikationskanäle. Dicke, graue Pfeile: Quantenkanäle.

6.2 3-Phasen-Commit-Protokoll: Koordinatorwahl mit W-Zustand

Im Terminierungsprotokoll des 3-Phasen-Commit-Protokolls aus Abschnitt 2.3.2 wird, nach der Idee von Skeen, ein neuer Koordinator gewählt, der durch Sammeln der Zustände der Teilnehmer die Transaktion entscheidet [Ske81]. Des Weiteren kann ein neu gewählter Koordinator ausfallen, sodass die Forderung besteht, die Wahl wieder und wieder ausführen zu können [BHG87; Ske81]. Im Folgenden wird untersucht, welche Vorteile eine Kombination des klassischen Terminierungsprotokolls mit den Konzepten des Quantencomputings hervorbringt. Genauer werden, wie in Abschnitt 5.1, die Eigenschaften der Verschränkung verwendet, um einen neuen Koordinator wählen zu können. So wird ebenfalls der W-Zustand aus Abschnitt 2.1.6 wie bei D'Hondt und Panangaden eingesetzt, um einen einzelnen Prozess per Zufall zu einem Koordinator zu wählen [DP04].

6.2.1 Umgebung und zusätzliche Mechanismen

Wie im herkömmlichen 3-Phasen-Commit-Protokoll in Abschnitt 2.3.3 wird nur von Ausfällen per Halten und nicht von Kommunikationsfehlern ausgegangen [Ske81; ST17]. Auch die Umgebung ist synchron, sodass Ausfälle durch Timeouts festgestellt werden können [Ske81].

Die Generierung und Verteilung eines W-Zustands auf die Teilnehmer wird von einer zentralen Einheit vorgenommen. Diese wird als ausfallsicher, beziehungsweise leicht ersetzbar angenommen. Alle Prozesse können, wenn nötig, mit der zentralen Einheit kommunizieren. Des Weiteren werden n Quantenkanäle, ausgehend von der zentralen Einheit, zu allen n Teilnehmern verwendet, wie beispielhaft in Abbildung 6.3 mit den grauen Quantenkanälen zwischen der zentralen Einheit und den Teilnehmern T1 bis T3 dargestellt. Diese sind ebenfalls ausfallsicher und verändern keine Nachrichten. Ein Quantenkanal für den Koordinator wird nicht benötigt.

Die zentrale Einheit empfängt in dieser Erweiterung ausschließlich Nachrichten von dem aktuellen Koordinator, wie im Kommunikationsverlauf in Abbildung 6.4 dargestellt. Der Koordinator schickt seine Nachrichten an die Teilnehmer und zusätzlich an die zentrale Einheit. Um Ausfälle des Koordinators detektieren zu können, die das Terminierungsprotokoll auslösen, verwendet die

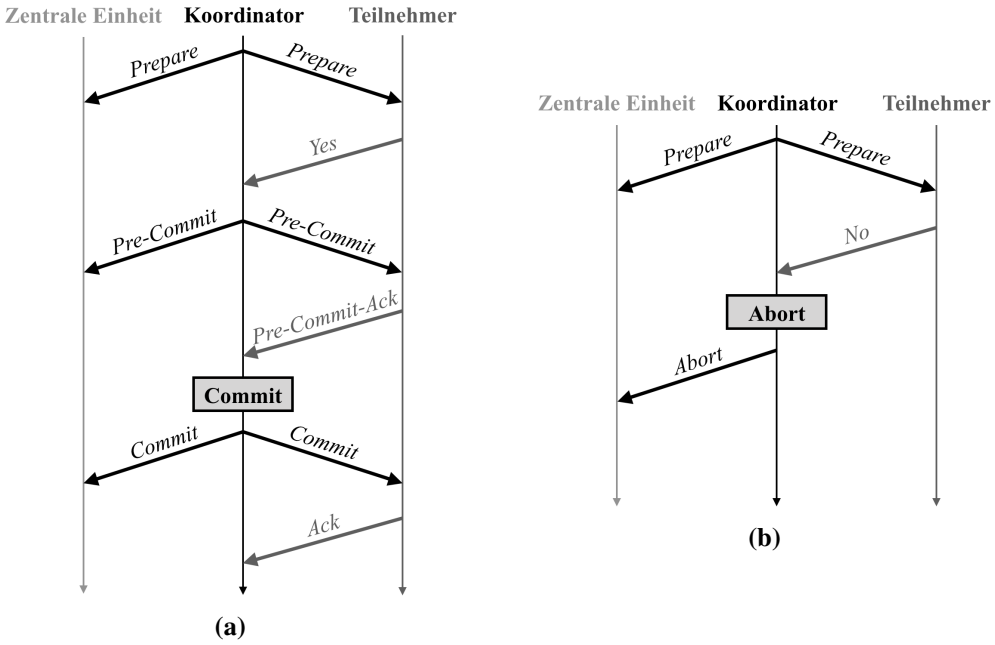

Abbildung 6.4: Kommunikationsverlauf des erweiterten 3-Phasen-Commit-Protokolls mit zentraler Einheit, orientiert an [BHG87; BN09; ST17]. (a) Commit einer Transaktion. (b) Abort einer Transaktion.

zentrale Einheit Timeouts, die jeweils nach Empfang der Nachrichten *Prepare* und *Pre-Commit* laufen. Diese errechnen sich aus der maximalen Antwortzeit der Teilnehmer sowie einem festgelegten Puffer an Bearbeitungszeit von Teilnehmern und Koordinator. Diese Timeouts werden ebenso innerhalb des Terminierungsprotokolls eingesetzt, um einen Ausfall des neuen Koordinators erkennen zu können. Ein weiterer Timeout wird für das Erkennen eines Ausfalls direkt nach der Wahl eines neuen Koordinators im Terminierungsprotokoll verwendet. Dieser Timeout wird entsprechend der Bearbeitungszeit und der Antwortzeit des Koordinators gewählt.

6.2.2 Ablauf des erweiterten 3-Phasen-Commit-Protokolls

Zu Beginn des erweiterten 3-Phasen-Commit-Protokolls sendet der Koordinator an alle Teilnehmer und die zentrale Einheit, neben der herkömmlichen *Prepare*-Nachricht, eine Liste aller Teilnehmer sowie der zentralen Einheit, ähnlich zu dem kooperativen Terminierungsprotokoll des 2-Phasen-Commit-Protokolls, Abschnitt 2.3.2 [BHG87]. Dadurch wird ermöglicht, dass sich sowohl die Teilnehmer als auch die zentrale Einheit untereinander kennen. Weiterhin läuft das Protokoll bezüglich des Nachrichtenaustauschs ab, wie bereits in Abschnitt 2.3.3 aufgezeigt. Die zentrale Einheit empfängt, wie in Abschnitt 6.2.1 erläutert, die Nachrichten des Koordinators und befindet sich ansonsten im Hintergrund, dargestellt in Abbildung 6.4.

Fällt der Koordinator nach dem Versenden der *Prepare* oder *Pre-Commit*-Nachrichten aus, stellt das, neben den Teilnehmern, auch die zentrale Einheit durch Timeouts fest, wie in Abschnitt 6.2.1 beschrieben, dargestellt in Abbildung 6.3. Nun wird das erweiterte Terminierungsprotokoll bei der

Algorithmus 6.1 Wahl eines Koordinators, orientiert an [DP04].

```
1: quBit ← empfangenes Qubit aus |W_n⟩
2: koordinator ← false
3: messErgebnis ← Messe(quBit)
4: if messErgebnis == 1 then
5:     koordinator ← true
6: end if
```

zentralen Einheit und den Teilnehmern aufgerufen. In den anderen Fällen, in denen der Koordinator ausfallen kann, greift die zentrale Einheit nicht ein, da hier die Teilnehmer selbst eine Entscheidung über die Transaktion treffen können, siehe Abschnitt 2.3.3 [BHG87; ST17].

6.2.3 Ablauf des erweiterten Terminierungsprotokolls

Nach Erkennung des Ausfalls des Koordinators in den in Abschnitt 6.2.2 genannten Fällen generiert die zentrale Einheit einen W-Zustand bestehend aus n Qubits, entsprechend der n Teilnehmer:

$$|W_n\rangle = \frac{1}{\sqrt{n}}(\overbrace{|\underbrace{0...001}_{n \text{ Qubits}}\rangle + |0...010\rangle + ... + |1...000\rangle}^{n \text{ Zustände}}) \qquad ,n : \#\text{ Teilnehmer} \tag{6.2}$$

Diese werden schließlich, wie bei dem Protokoll von D'Hondt und Panangaden, auf die Teilnehmer verteilt, sodass jeder Teilnehmer genau ein Qubit des W-Zustands erhält, wie in Abbildung 6.3 mit den Qubits q_1 bis q_3 für die drei Teilnehmer T1 bis T3 gezeigt [DP04]. Die Teilnehmer bleiben zunächst blockiert, bis sie die Qubits der zentralen Einheit empfangen. Bei Empfang der Qubits führen die Teilnehmer den Algorithmus 6.1 aus. Hierbei sind sie initial weiterhin in der Rolle der Teilnehmer, siehe Zeile 2. Messen sie ihr Qubit in Zeile 3 in der Standardbasis, erhalten sie entweder eine 0 oder eine 1. Während genau ein Teilnehmer eine 1 misst, messen alle anderen Teilnehmer eine 0. Die, die eine 0 messen bleiben weiterhin Teilnehmer. Der Prozess, der eine 1 misst, ist von nun an der neue Koordinator und führt das Terminierungsprotokoll an, siehe Zeile 4 und Zeile 5 des Algorithmus 6.1.

Der neu erwählte Koordinator sendet nun an alle Teilnehmer und die zentrale Einheit *State-Request*-Nachrichten, sammelt die Antworten der Teilnehmer ein und entscheidet die Transaktion entsprechend, wie bereits in Abschnitt 2.3.4 beschrieben [BHG87; Ske81]. Auch hier befindet sich die zentrale Einheit, bis auf das Empfangen der Nachrichten des Koordinators, im Hintergrund.

Nach Verteilen der Qubits und nach Empfangen der Nachrichten *State-Request* und *Pre-Commit*, siehe Abschnitt 2.3.4, misst die zentrale Einheit die Zeit, um einen Ausfall des neu gewählten Koordinators zu erkennen. In allen drei Fällen, können die Teilnehmer nicht selbst eine Entscheidung über die Transaktion treffen.

Trifft ein Ausfall des Koordinators entsprechend den oben genannten Fällen zu, wird bei der zentralen Einheit ein Timeout ausgelöst. Daraufhin generiert sie einen neuen W-Zustand mit $n - 1$ Qubits, siehe Gleichung (6.2). Die Anzahl der Qubits entspricht den noch übrig gebliebenen Teilnehmern.

Die Qubits des W-Zustands werden wiederum auf die $n-1$ Teilnehmer verteilt, wie in Abbildung 6.3 dargestellt. Diese führen erneut den Algorithmus 6.1 aus und messen die Qubits. Per Zufall wird schließlich ein neuer Koordinator gewählt, der das erweiterte Terminierungsprotokoll erneut von Beginn an ausführt. Eine Neuwahl kann solange wiederholt ausgeführt werden, bis schließlich noch zwei Teilnehmer aktiv sind. Ist nur noch ein Teilnehmer aktiv, sendet ihm die zentrale Einheit ein einzelnes Qubit mit dem Zustand $|1\rangle$. So wird er schließlich Koordinator und entscheidet die Transaktion selbst.

6.2.4 Vorteile bei Verwendung des W-Zustands

Durch die Verwendung des W-Zustands als Mittel zur Wahlentscheidung ergibt sich der Vorteil, dass keine aufwändige Kommunikation zwischen den Teilnehmern aufgebracht werden muss. Die Teilnehmer selbst müssen keinen Konsens untereinander finden, um einen Koordinator zu wählen.

Des Weiteren wird der W-Zustand erst zu dem Zeitpunkt generiert, zu dem er auch benötigt wird. Dadurch muss die Stabilität des Quantenzustands nur für eine relativ kurze Zeit bis zur Messung aufrecht erhalten werden, siehe Abschnitt 2.1.11 [Hom18]. Zudem werden durch die Einbindung der zentralen Einheit zwar mehr Nachrichten benötigt, jedoch ist die Anzahl der Quantenkanäle gering. Würden die Prozesse das Generieren und Verteilen der W-Zustande selbst durchführen, würden $n(n+1)$ Quantenkanäle benötigt. Wobei n die Anzahl der Teilnehmer ist und der erste Koordinator hinzu addiert wird.

Hinzu kommt, dass durch die Eigenschaften des W-Zustands höchstens ein Teilnehmer Koordinator wird, sodass ein Vorkommen von mehreren nicht möglich ist. Ein Ausfall des Koordinators wird durch die synchrone Umgebung zeitnah von der zentralen Einheit erkannt, sodass die Teilnehmer nicht lange blockiert sind.

6.2.5 Mögliche Probleme

Obwohl, ein W-Zustand im erweiterten Protokoll bereits kurz nach Erzeugung von den Teilnehmern gemessen wird, könnte es in der Realität dennoch sein, dass der Zustand nicht lange genug aufrecht gehalten werden kann, wie in Abschnitt 2.1.11 erläutert [Hom18].

Zudem ist das Wählen eines neuen Koordinators anhand der Größe der Identifikationsnummern der Prozesse, wie es Skeen und Bernstein et al. vorschlagen, ebenfalls eine effiziente Lösung [BHG87; Ske81]. Dabei kommt hinzu, dass diese allein durch klassische Computer umgesetzt werden kann.

Des Weiteren werden für das erweiterte 3-Phasen-Commit-Protokoll, durch die Einbindung der zentralen Einheit eine geringe Anzahl an Nachrichten mehr benötigt, wie in Abbildung 6.4 dargestellt. So werden aus den $6n$ Nachrichten für einen Durchlauf des Protokolls ohne Ausfälle, wie in Abschnitt 2.3.3 gezeigt, $6n+3$ Nachrichten [BHG87]. Dabei ist auch hier n die Anzahl der Teilnehmer. Zu den hinzukommenden Nachrichten gehören die Nachrichten des Koordinators, die zusätzlich an die zentrale Einheit gesendet werden. Mit Ausfällen des Koordinators erhöht sich die Anzahl der Nachrichten an die zentrale Einheit entsprechend. Bei Aufruf des erweiterten Terminierungsprotokolls muss unter anderem die Verteilung der Qubits des W-Zustands berücksichtigt werden, siehe Abbildung 6.3. Weiterhin werden alle Nachrichten des neuen Koordinators

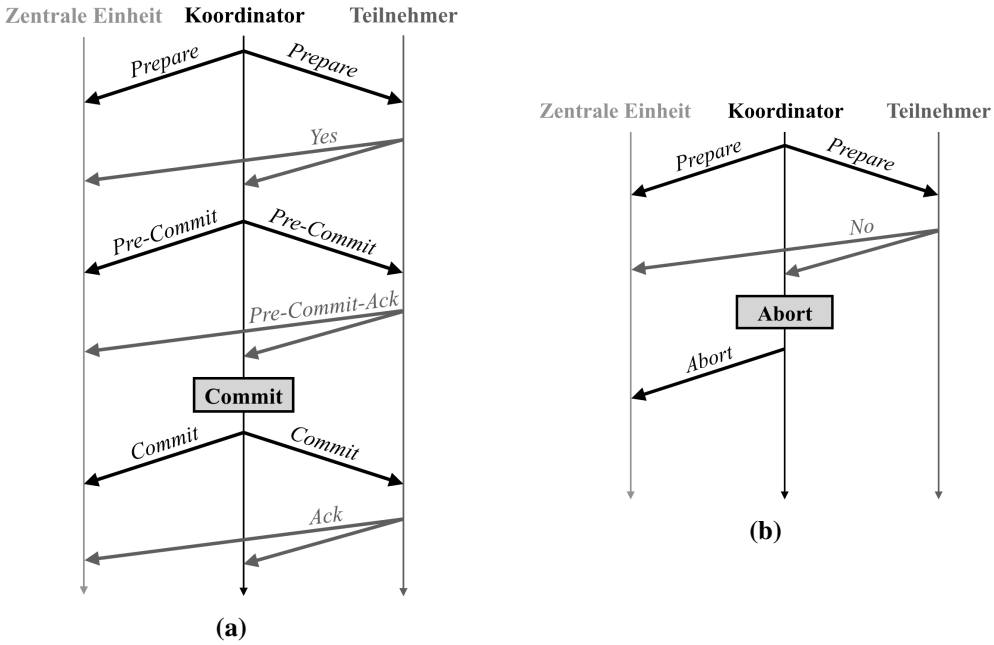

Abbildung 6.5: Kommunikationsverlauf des erweiterten 3-Phasen-Commit-Protokolls mit zentraler Einheit, die von allen Prozessen Nachrichten empfängt, orientiert an [BHG87; BN09; ST17]. (a) Commit einer Transaktion. (b) Abort einer Transaktion.

ebenso an die zentrale Einheit gesendet. So werden bei einmaliger Ausführung des erweiterten Terminierungsprotokolls ohne Einbeziehen der Wahl, bei dem nur der Koordinator ausgefallen ist, mindestens $4n - 2$ Nachrichten, anstatt $4(n - 1)$, benötigt, siehe Abschnitt 2.3.4 [BHG87].

6.2.6 Zusätzliche Erweiterung: Berücksichtigung von Teilnehmerausfällen

Im den bisherigen Abläufen des erweiterten 3-Phasen-Commit-Protokolls und Terminierungsprotokolls werden nur Ausfälle des aktuellen Koordinators berücksichtigt. Um ebenso Ausfälle von einzelnen Teilnehmern bei der Generierung und Verteilung der W-Zustände berücksichtigen zu können, wird eine höhere Anzahl an Nachrichten benötigt, wie im Kommunikationsverlauf in Abbildung 6.5 dargestellt.

So muss nicht nur der Koordinator seine Nachrichten zusätzlich an die zentrale Einheit senden, sondern auch die Teilnehmer müssen ihre Antworten an sie übermitteln, siehe Abbildung 6.5. Auch hierfür verwendet die zentrale Einheit nach Erhalt jeder Nachricht eines Teilnehmers einen Timeout. Dieser ergibt sich aus der Antwortzeit des Koordinators und den Bearbeitungszeiten des Koordinators und des Teilnehmers. Wird während des Wartens auf die Antwort eines Teilnehmers bei der zentralen Einheit ein Timeout ausgelöst, gilt der Teilnehmer als ausgefallen. Werden erneut Nachrichten von ihm empfangen, gilt er wieder als aktiv. Ebenso werden Teilnehmer behandelt, die in der Abstimmungsphase mit einem *No* antworten und die Transaktion mit einem Abort abschließen, wie in Abbildung 6.5b gezeigt.

Teilnehmer die bei Beginn des Terminierungsprotokolls nicht aktiv sind, werden auch nach ihrer Wiederherstellung nicht in dessen Ablauf miteinbezogen, wie bereits in Abschnitt 2.3.4 beschrieben [BHG87]. Durch die Berücksichtigung von Teilnehmerausfällen kann die Anzahl der Qubits des W-Zustands entsprechend gewählt werden, sodass nur aktiven Teilnehmern ein Qubit zugewiesen wird. So wird vermieden, dass Qubits an Teilnehmer verteilt werden, die bereits ausgefallen sind. Dadurch ist garantiert, dass einer der aktiven Teilnehmer als Koordinator gewählt wird. Solche Wahlphasen, die in diesen Fällen bereits im Vorhinein erfolglos sind, können somit vermieden werden.

6.3 Zusammenfassende Erkenntnisse

Wie in den Abschnitten 6.1.3 und 6.1.4 gezeigt, ist die Verwendung des GHZ-Zustands für das Deblockieren des 2-Phasen-Commit-Protokolls und dessen kooperatives Terminierungsprotokoll nur unter bestimmten Gegebenheiten hilfreich.

Da das Commit oder Abort einer Transaktion unwiderruflich ist, ist das Zurückziehen einer Entscheidung bei Unstimmigkeiten nicht möglich [Ske81]. Mit dem Zurückziehen einer Transaktion könnten die genannten Inkonsistenzen, die in Abschnitt 6.1 aufgezeigt werden, jedoch behoben werden. Dies ist im Allgemeinen jedoch nicht gewollt. Es zeigt sich, dass der GHZ-Zustand mit Messung in der Standardbasis eine Deblockierung mit stets sichergestellter Atomarität nicht ermöglichen kann.

Auch wenn das 3-Phasen-Commit-Protokoll meist nur Anwendung in der Theorie hat, zeigt sich in Abschnitt 6.2, dass Konzepte der Quantenmechanik verwendet werden können, um bestimmte Anforderungen des Protokolls zu erfüllen [BHG87; ST17]. So kann die Wahl eines neuen Koordiantors für das Terminierungsprotokoll mit Hilfe des W-Zustands durchgeführt werden. Verglichen mit einem herkömmlichen Wahlverfahren mittels Abstimmung, wird durch die Verschränkung keine Kommunikation zwischen den Teilnehmern benötigt, um einen Konsens zu finden.

Zusätzliche Kommunikation kann in der präsentierten Erweiterung in Abschnitt 6.2 nicht vollständig vermieden werden. Zudem gibt es bereits andere Wahlverfahren mit klassischen Computern, die in der gegebenen Umgebung geringen Aufwand benötigen, wie bereits in Abschnitt 6.2.5 angesprochen [BHG87; Ske81].

7 Zusammenfassung und Ausblick

In dieser Arbeit wurde zum einen untersucht, ob Verschränkung die Ursache für den möglichen Speedup von Quantencomputern gegenüber klassischen Computern ist. Zum anderen wurden Konsensusprotokolle für Quantencomputer entwickelt und mit ihren klassischen Pendants verglichen. Dazu wurden zunächst die Grundlagen der Quanteninformatik aufgezeigt. Anschließend wurde die Aufgabe von Konsensusprotokollen in verteilten Systemen erläutert. Als ein beispielhaftes Konsensusprotokoll wurde Paxos [Lam98] näher betrachtet und auf dessen Eigenschaften eingegangen. Zugehörig wurden verwandte Arbeiten über Konsensusprotokolle für Quantencomputer präsentiert. Diese zeigten, dass durch die Verwendung von Quantencomputing Vorteile, wie Kommunikationseinsparungen, möglich sein können [BH05; CKS10; DP04]. Als eine spezielle Unterkategorie von Konsensusprotokollen wurde das blockierende 2-Phasen-Commit-Protokoll [Gra78] und das 3-Phasen-Commit-Protokoll [Ske81] sowie deren zugehörige Terminierungsprotokolle präsentiert.

Anschließend wurden Einblicke in die möglichen Errungenschaften und den aktuellen Stand des Quantencomputings gegeben. Dabei wurden verschiedene Gebiete vorgestellt, in denen Quantencomputer große Bereicherungen sein könnten. Dennoch liegt die Anwendbarkeit von Quantencomputern auf größere Probleme noch in weiter Ferne [Bro19; Nat19; Pre18]. So ist auch bisher nicht erschlossen, ob Quantencomputer im Allgemeinen tatsächlich leistungsfähiger sind als klassische Computer [NC11; RP11].

Darauffolgend wurde auf die Frage des Grunds für den möglichen Speedup gegenüber klassischen Algorithmen eingegangen. So zeigte sich, dass Algorithmen mit reinen Zuständen ein gewisses Maß an Verschränkung benötigen, um einen einen exponentiellen Speedup zu erhalten [JL03; RP11]. Dennoch wird angenommen, dass Verschränkung nicht als einzige Ursache für den Speedup gesehen werden sollte [DJ07; H09; Ved10].

Schließlich wurden die entwickelten Konsensusprotokolle präsentiert. Für die erste Erweiterung von Paxos wurde der W-Zustand für die Wahl eines einzelnen Proposers verwendet. Diese Erweiterung zeigte insbesondere eine Verbesserung der Fortschrittlichkeit des Protokolls. In der zweiten Erweiterung wurde Superposition zur Vergabe der Rundennummer verwendet. Dabei wurde gezeigt, dass durch die Erweiterung von einem erweiterten Fehlermodell ausgegangen werden kann [CKS10]. Bei beiden Erweiterungen zeigte sich, dass zusätzliche Mechanismen verwendet werden müssen, um byzantinische Fehler tolerieren zu können. Anschließend wurden zwei Erweiterungen für das 2-Phasen-Commit-Protokoll präsentiert, die jeweils einen GHZ-Zustand verwendeten. Es zeigte sich, dass sowohl im 2-Phasen-Commit-Protokoll, als auch im kooperativen Terminierungsprotokoll, die Verwendung des GHZ-Zustands ein Blockieren der Teilnehmer nicht verhindern kann. Zuletzt wurde das 3-Phasen-Commit-Protokoll mit Hilfe des W-Zustands für die Wahl eines neuen Koordinators gewählt. Es zeigte sich, dass durch die Verschränkung zusätzliche Kommunikation zwischen den Teilnehmern für eine Wahl vermieden werden kann.

Zusammengefasst konnte gezeigt werden, dass W-Zustände sich besonders für die zufällige Wahl eines Leaders eignen, ohne dass ein zusätzlicher Austausch zwischen den Wählern stattfinden muss, wie bereits D'Hondt und Panangaden [DP04] festgestellt haben. Der GHZ-Zustand ist insbesondere für eine Konsensfindung geeignet für den Fall, dass der Wert, auf den sich geeinigt werden soll, nicht von großer Relevanz ist und zufällig gewählt werden kann. Die Blockierung des 2-Phasen-Commit-Protokolls konnte mit Hilfe des GHZ-Zustands in der angewandten Weise nicht gelöst werden.

Ausblick

Im Hinblick auf diese Arbeit sollte in Zukunft eine mögliche Umsetzung der erweiterten Protokolle erarbeitet werden. Durch die beispielhaften Implementierungen könnten sich weitere Eigenschaften der Erweiterungen zeigen. Zudem werden so mögliche Herausforderungen einer Umsetzung hervorgehoben, die in einer rein theoretischen Konzipierung nicht ersichtlich sind. Es werden sich schließlich Machbarkeit und Aufwand der Implementierungen der erweiterten Protokolle zeigen.

Die aufgezeigten Erweiterungen basieren auf Zuständen in Superposition und im Speziellen auf W- und GHZ-Zuständen. Diese Zustände sind heutzutage jedoch noch sehr instabil, wodurch die Einsetzbarkeit der Protokolle sehr eingeschränkt ist [Hom18]. Durch Verbesserungen bezüglich der Stabilität der Quantenzustände wird auch die Einsetzbarkeit der Protokolle erhöht. Im Allgemeinen werden Fortschritte in der Quantenmechanik und der Quanteninformatik auch zu Fortschritten bei Konsensusprotokollen für Quantencomputer führen.

Literaturverzeichnis

[AAB+19] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. Brandao, D. A. Buell et al. „Quantum supremacy using a programmable superconducting processor“. In: *Nature* 574.7779 (2019), S. 505–510. DOI: 10.1038/s41586-019-1666-5 (zitiert auf S. 43, 44).

[Aar10] S. Aaronson. „BQP and the polynomial hierarchy“. In: *Proceedings of the 42nd ACM symposium on Theory of computing - STOC '10* (2010). DOI: 10.1145/1806689.1806711. URL: http://dx.doi.org/10.1145/1806689.1806711 (zitiert auf S. 28).

[ACD16] A. Ailijiang, A. Charapko, M. Demirbas. „Consensus in the Cloud: Paxos Systems Demystified“. In: *2016 25th International Conference on Computer Communication and Networks (ICCCN)*. Aug. 2016, S. 1–10. DOI: 10.1109/ICCCN.2016.7568499 (zitiert auf S. 58).

[Asp02] J. Aspnes. „Randomized protocols for asynchronous consensus“. In: *CoRR* cs.DS/0209014 (2002). URL: http://arxiv.org/abs/cs.DS/0209014 (zitiert auf S. 28, 33).

[BBKM04] E. Biham, G. Brassard, D. Kenigsberg, T. Mor. „Quantum computing without entanglement“. In: *Theoretical Computer Science* 320.1 (2004), S. 15–33. DOI: https://doi.org/10.1016/j.tcs.2004.03.041 (zitiert auf S. 25, 26, 47).

[BH05] M. Ben-Or, A. Hassidim. „Fast Quantum Byzantine Agreement“. In: *Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing*. STOC '05. Baltimore, MD, USA: ACM, 2005, S. 481–485. ISBN: 1-58113-960-8. DOI: 10.1145/1060590.1060662. URL: http://doi.acm.org/10.1145/1060590.1060662 (zitiert auf S. 13, 33, 34, 52, 54, 69).

[BHG87] P. A. Bernstein, V. Hadzilacos, N. Goodman. *Concurrency Control and Recovery in Database Systems*. Addison-Wesley, 1987. ISBN: 0-201-10715-5. URL: http://research.microsoft.com/en-us/people/philbe/ccontrol.aspx (zitiert auf S. 34–42, 59–61, 63–68).

[BN09] P. A. Bernstein, E. Newcomer. *Principles of Transaction Processing*. 2nd. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2009. ISBN: 1558606238, 9781558606234 (zitiert auf S. 34, 35, 37, 39, 61, 64, 67).

[Bro19] M. Brooks. „Beyond quantum supremacy: the hunt for useful quantum computers.“ In: *Nature* 574.7776 (2019), S. 19. DOI: 10.1038/d41586-019-02936-3 (zitiert auf S. 13, 43, 44, 69).

[CEP+18] P. J. Coles, S. Eidenbenz, S. Pakin, A. Adedoyin, J. Ambrosiano, P. Anisimov, W. Casper, G. Chennupati, C. Coffrin, H. Djidjev et al. „Quantum algorithm implementations for beginners“. In: *arXiv preprint arXiv:1804.03719* (2018) (zitiert auf S. 26).

[CGR07] T. D. Chandra, R. Griesemer, J. Redstone. „Paxos made live: an engineering perspective“. In: *Proceedings of the twenty-sixth annual ACM symposium on Principles of distributed computing*. ACM. 2007, S. 398–407 (zitiert auf S. 29–31, 58).

[CKS10] B. S. Chlebus, D. R. Kowalski, M. Strojnowski. „Scalable Quantum Consensus for Crash Failures“. In: *Distributed Computing*. Hrsg. von N. A. Lynch, A. A. Shvartsman. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, S. 236–250. ISBN: 978-3-642-15763-9 (zitiert auf S. 13, 32, 33, 53–58, 69).

[CL+99] M. Castro, B. Liskov et al. „Practical Byzantine fault tolerance“. In: *OSDI*. Bd. 99. 1999. 1999, S. 173–186 (zitiert auf S. 53, 57).

[CMS89] B. Chor, M. Merritt, D. B. Shmoys. „Simple constant-time consensus protocols in realistic failure models“. In: *Journal of the ACM (JACM)* 36.3 (1989), S. 591–614 (zitiert auf S. 33).

[CV17] C. Cachin, M. Vukolić. *Blockchain Consensus Protocols in the Wild*. 2017. arXiv: 1707.01873 [cs.DC] (zitiert auf S. 53, 57).

[CY97] I. L. Chuang, Y. Yamamoto. „Creation of a persistent quantum bit using error correction“. In: *Phys. Rev. A* 55 (1 Jan. 1997), S. 114–127. DOI: 10.1103/PhysRevA.55.114. URL: https://link.aps.org/doi/10.1103/PhysRevA.55.114 (zitiert auf S. 26).

[DH96] C. Dürr, P. Hoyer. *A Quantum Algorithm for Finding the Minimum*. Techn. Ber. quant-ph/9607014. Juli 1996. URL: http://cds.cern.ch/record/307291 (zitiert auf S. 27).

[DJ07] S. Ding, Z. Jin. „Review on the study of entanglement in quantum computation speedup“. In: *Chinese Science Bulletin* 52.16 (Aug. 2007), S. 2161–2166. ISSN: 1861-9541. DOI: 10.1007/s11434-007-0324-8. URL: https://doi.org/10.1007/s11434-007-0324-8 (zitiert auf S. 45, 47, 69).

[DLL00] R. De Prisco, B. Lampson, N. Lynch. „Revisiting the PAXOS algorithm“. In: *Theoretical Computer Science* 243.1-2 (2000), S. 35–91. DOI: 10.1016/S0304-3975(00)00042-6. URL: https://doi.org/10.1016/S0304-3975(00)00042-6 (zitiert auf S. 29–31, 55–58).

[DP04] E. D'Hondt, P. Panangaden. „The computational power of the W and GHZ states“. In: *arXiv preprint quant-ph/0412177* (2004) (zitiert auf S. 13, 31, 32, 34, 49–52, 60, 63, 65, 69, 70).

[DSC08] A. Datta, A. Shaji, C. M. Caves. „Quantum Discord and the Power of One Qubit“. In: *Physical Review Letters* 100.5 (Feb. 2008). ISSN: 1079-7114. DOI: 10.1103/physrevlett.100.050502. URL: http://dx.doi.org/10.1103/PhysRevLett.100.050502 (zitiert auf S. 47).

[Dür18] D. F. Dürr. „Distributed Systems - Transaction Processing“. Universitätsvorlesung. Institut für Parallele und Verteilte Systeme (IPVS) - Universität Stuttgart, 2018 (zitiert auf S. 35, 39).

[EJ98] A. Ekert, R. Jozsa. „Quantum algorithms: entanglement–enhanced information processing“. In: *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* 356.1743 (1998), S. 1769–1782. URL: https://doi.org/10.1098/rsta.1998.0248 (zitiert auf S. 46).

[EPR35] A. Einstein, B. Podolsky, N. Rosen. „Can quantum-mechanical description of physical reality be considered complete?“ In: *Physical review* 47.10 (Mai 1935), S. 777–780. DOI: 10.1103/PhysRev.47.777. URL: https://link.aps.org/doi/10.1103/PhysRev.47.777 (zitiert auf S. 22).

[Fey82] R. P. Feynman. „Simulating physics with computers“. In: *International journal of theoretical physics* 21.6 (1982), S. 467–488 (zitiert auf S. 13, 46).

[FGG14] E. Farhi, J. Goldstone, S. Gutmann. „A quantum approximate optimization algorithm“. In: *arXiv preprint arXiv:1411.4028* (2014) (zitiert auf S. 26).

[FLP85] M. J. Fischer, N. A. Lynch, M. S. Paterson. „Impossibility of Distributed Consensus with One Faulty Process“. In: *Journal of the Association for Computing Machinery* 32.2 (1985), S. 374–382 (zitiert auf S. 29, 31, 50).

[GFE09] D. Gross, S. T. Flammia, J. Eisert. „Most Quantum States Are Too Entangled To Be Useful As Computational Resources“. In: *Phys. Rev. Lett.* 102 (19 Mai 2009), S. 190501. DOI: 10.1103/PhysRevLett.102.190501. URL: https://link.aps.org/doi/10.1103/PhysRevLett.102.190501 (zitiert auf S. 46).

[GHZ07] D. M. Greenberger, M. A. Horne, A. Zeilinger. *Going Beyond Bell's Theorem*. 2007. arXiv: 0712.0921 [quant-ph] (zitiert auf S. 22).

[Gib19] E. Gibney. „Hello quantum world! Google publishes landmark quantum supremacy claim.“ In: *Nature* 574.7779 (2019), S. 461. DOI: 10.1038/d41586-019-03213-z (zitiert auf S. 43, 44).

[GIN18] L. Gyongyosi, S. Imre, H. V. Nguyen. „A Survey on Quantum Channel Capacities“. In: *IEEE Communications Surveys Tutorials* 20.2 (2018), S. 1149–1205. DOI: 10.1109/COMST.2017.2786748 (zitiert auf S. 23, 24, 56).

[GKM09] C. Gavoille, A. Kosowski, M. Markiewicz. „What can be observed locally?“ In: *International Symposium on Distributed Computing*. Springer. 2009, S. 243–257. ISBN: 978-3-642-04355-0. URL: https://doi.org/10.1007/978-3-642-04355-0_26 (zitiert auf S. 32, 34).

[GM19] G. G. Guerreschi, A. Matsuura. „QAOA for Max-Cut requires hundreds of qubits for quantum speed-up“. In: *Scientific reports* 9.1 (2019), S. 6903. DOI: 10.1038/s41598-019-43176-9 (zitiert auf S. 44).

[Gra78] J. N. Gray. „Notes on data base operating systems“. In: *Operating Systems: An Advanced Course*. Hrsg. von R. Bayer, R. M. Graham, G. Seegmüller. Berlin, Heidelberg: Springer Berlin Heidelberg, 1978, S. 393–481. ISBN: 978-3-540-35880-0. DOI: 10.1007/3-540-08755-9_9. URL: https://doi.org/10.1007/3-540-08755-9_9 (zitiert auf S. 34, 35, 59, 69).

[Gro96] L. K. Grover. „A fast quantum mechanical algorithm for database search“. In: *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing - STOC '96* (1996). DOI: 10.1145/237814.237866. URL: http://dx.doi.org/10.1145/237814.237866 (zitiert auf S. 27, 47).

[H09] R. Horodecki, P. Horodecki, M. Horodecki, K. Horodecki. „Quantum entanglement“. In: *Reviews of Modern Physics* 81.2 (Juni 2009), S. 865–942. ISSN: 1539-0756. DOI: 10.1103/revmodphys.81.865. URL: http://dx.doi.org/10.1103/revmodphys.81.865 (zitiert auf S. 46, 47, 69).

[Haa15] Haale (Saale) : Deutsche Akademie der Naturforscher Leopoldina e.V. - Nationale Akademie der Wissenschaften - München : acatech - Deutsche Akademie der Technikwissenschaften - Mainz : Union der deutschen Akademien der Wissenschaften e.V. „Perspektiven der Quantentechnologien (2015)“. In: (2015) (zitiert auf S. 13, 43, 44).

[Hom18] M. Homeister. *Quantum Computing verstehen*. Springer Vieweg, Wiesbaden, 2018. ISBN: 9783658228842, 9783658228835. DOI: 10.1007/978-3-658-22884-2 (zitiert auf S. 13, 15–28, 45, 58, 66, 70).

[JL03] R. Jozsa, N. Linden. „On the role of entanglement in quantum-computational speed-up“. In: *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* 459.2036 (Aug. 2003), S. 2011–2032. ISSN: 1471-2946. DOI: 10.1098/rspa.2002.1097. URL: http://dx.doi.org/10.1098/rspa.2002.1097 (zitiert auf S. 45–47, 69).

[Joz98] R. Jozsa. *Pages 369-379 in The Geometric Universe edited by S. Huggett, L. Mason, KP Tod, ST Tsou and N. Woodhouse*. 1998 (zitiert auf S. 46).

[KM04] V. M. Kendon, W. J. Munro. *Entanglement and its Role in Shor's Algorithm*. 2004. arXiv: quant-ph/0412140 [quant-ph] (zitiert auf S. 46).

[Lam01] L. Lamport. „Paxos made simple“. In: *ACM Sigact News* 32.4 (2001), S. 18–25 (zitiert auf S. 28–31, 49, 51–56, 58).

[Lam06] L. Lamport. „Fast paxos“. In: *Distributed Computing* 19.2 (2006), S. 79–103. ISSN: 1432-0452. DOI: 10.1007/s00446-006-0005-x. URL: https://doi.org/10.1007/s00446-006-0005-x (zitiert auf S. 31, 49).

[Lam96] B. Lampson. „How to Build a Highly Available System Using Consensus“. In: Springer-Verlag, Okt. 1996. URL: https://www.microsoft.com/en-us/research/publication/how-to-build-a-highly-available-system-using-consensus/ (zitiert auf S. 13, 14, 28, 30, 31, 58).

[Lam98] L. Lamport. „The part-time parliament“. In: *ACM Transactions on Computer Systems (TOCS)* 16.2 (1998), S. 133–169. DOI: 10.1145/279227.279229 (zitiert auf S. 14, 29, 49, 59, 69).

[LBAW08] B. P. Lanyon, M. Barbieri, M. P. Almeida, A. G. White. „Experimental Quantum Computing without Entanglement“. In: *Phys. Rev. Lett.* 101 (20 Nov. 2008), S. 200501. DOI: 10.1103/PhysRevLett.101.200501. URL: https://link.aps.org/doi/10.1103/PhysRevLett.101.200501 (zitiert auf S. 47).

[Llo99] S. Lloyd. „Quantum search without entanglement“. In: *Phys. Rev. A* 61 (1 Dez. 1999), S. 010301. DOI: 10.1103/PhysRevA.61.010301. URL: https://link.aps.org/doi/10.1103/PhysRevA.61.010301 (zitiert auf S. 47).

[LP01] N. Linden, S. Popescu. „Good Dynamics versus Bad Kinematics: Is Entanglement Needed for Quantum Computation?“ In: *Physical Review Letters* 87.4 (Juli 2001). ISSN: 1079-7114. DOI: 10.1103/physrevlett.87.047901. URL: http://dx.doi.org/10.1103/PhysRevLett.87.047901 (zitiert auf S. 47).

[LSP82] L. Lamport, R. Shostak, M. Pease. „The Byzantine generals problem“. In: *ACM Transactions on Programming Languages and Systems (TOPLAS)* 4.3 (1982), S. 382–401 (zitiert auf S. 29).

[MDLH18] A. McCaskey, E. Dumitrescu, D. Liakh, T. Humble. „Hybrid Programming for Near-Term Quantum Computing Systems“. In: *2018 IEEE International Conference on Rebooting Computing (ICRC)*. Nov. 2018, S. 1–12. DOI: 10.1109/ICRC.2018.8638598 (zitiert auf S. 26).

[Mer07] N. D. Mermin. *Quantum Computer Science: An Introduction*. Cambridge University Press, 2007. ISBN: 9780511813870. DOI: 10.1017/CBO9780511813870 (zitiert auf S. 16).

[Mey00] D. A. Meyer. „Sophisticated Quantum Search Without Entanglement“. In: *Phys. Rev. Lett.* 85 (9 Aug. 2000), S. 2014–2017. DOI: 10.1103/PhysRevLett.85.2014. URL: https://link.aps.org/doi/10.1103/PhysRevLett.85.2014 (zitiert auf S. 47).

[MJ13] H. Meling, L. Jehl. „Tutorial summary: Paxos explained from scratch“. In: *International Conference On Principles Of Distributed Systems*. Springer. 2013, S. 1–10. ISBN: 978-3-319-03850-6. URL: https://doi.org/10.1007/978-3-319-03850-6_1 (zitiert auf S. 30, 53, 56).

[ML83] C. Mohan, B. Lindsay. „Efficient Commit Protocols for the Tree of Processes Model of Distributed Transactions“. In: *Proceedings of the Second Annual ACM Symposium on Principles of Distributed Computing*. PODC '83. Montreal, Quebec, Canada: ACM, 1983, S. 76–88. ISBN: 0-89791-110-5. DOI: 10.1145/800221.806711. URL: http://doi.acm.org/10.1145/800221.806711 (zitiert auf S. 34–37).

[Mon16] A. Montanaro. „Quantum algorithms: an overview“. In: *npj Quantum Information* 2 (2016), S. 15023. DOI: 10.1038/npjqi.2015.23. URL: https://doi.org/10.1038/npjqi.2015.23 (zitiert auf S. 27, 45, 46).

[MRBA16] J. R. McClean, J. Romero, R. Babbush, A. Aspuru-Guzik. „The theory of variational hybrid quantum-classical algorithms“. In: *New Journal of Physics* 18.2 (Feb. 2016), S. 023023. DOI: 10.1088/1367-2630/18/2/023023 (zitiert auf S. 26).

[Nat19] National Academies of Sciences, Engineering, and Medicine. *Quantum Computing: Progress and Prospects*. Hrsg. von E. Grumbling, M. Horowitz. Washington, DC: The National Academies Press, 2019. ISBN: 978-0-309-47969-1. DOI: 10.17226/25196. URL: https://www.nap.edu/catalog/25196/quantum-computing-progress-and-prospects (zitiert auf S. 13, 43, 44, 69).

[NC11] M. A. Nielsen, I. L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. 10th. New York, NY, USA: Cambridge University Press, 2011. ISBN: 1107002176, 9781107002173 (zitiert auf S. 13, 15, 16, 18–21, 23–26, 28, 45, 69).

[PGN+19] E. Pednault, J. A. Gunnels, G. Nannicini, L. Horesh, R. Wisnieff. *Leveraging Secondary Storage to Simulate Deep 54-qubit Sycamore Circuits*. 2019. arXiv: 1910.09534 [quant-ph] (zitiert auf S. 43).

[PMS+14] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, J. L. O'brien. „A variational eigenvalue solver on a photonic quantum processor“. In: *Nature communications* 5 (2014), S. 4213. DOI: 10.1038/ncomms5213. URL: https://doi.org/10.1038/ncomms5213 (zitiert auf S. 26).

[Pre18] J. Preskill. „Quantum Computing in the NISQ era and beyond“. In: *Quantum* 2 (Aug. 2018), S. 79. ISSN: 2521-327X. DOI: 10.22331/q-2018-08-06-79. URL: https://doi.org/10.22331/q-2018-08-06-79 (zitiert auf S. 13, 26, 27, 43, 44, 69).

[RK98] P. K. Reddy, M. Kitsuregawa. „Reducing the blocking in two-phase commit protocol employing backup sites“. In: *Proceedings. 3rd IFCIS International Conference on Cooperative Information Systems (Cat. No.98EX122)*. Aug. 1998, S. 406–415. DOI: 10.1109/COOPIS.1998.706315 (zitiert auf S. 62).

[RP00] E. Rieffel, W. Polak. „An introduction to quantum computing for non-physicists“. In: *ACM Computing Surveys (CSUR)* 32.3 (2000), S. 300–335. DOI: 10.1145/367701.367709 (zitiert auf S. 19–23, 27, 45).

[RP11] E. Rieffel, W. Polak. *Quantum Computing A Gentle Introduction*. MIT Press, März 2011. ISBN: 978-0-262-01506-6 (zitiert auf S. 13, 15–18, 20–27, 43–47, 58, 69).

[RSA78] R. L. Rivest, A. Shamir, L. Adleman. „A Method for Obtaining Digital Signatures and Public-key Cryptosystems“. In: *Commun. ACM* 21.2 (Feb. 1978), S. 120–126. ISSN: 0001-0782. DOI: 10.1145/359340.359342. URL: http://doi.acm.org/10.1145/359340.359342 (zitiert auf S. 27, 43).

[SAC+18] M. Suchara, Y. Alexeev, F. Chong, H. Finkel, H. Hoffmann, J. Larson, J. Osborn, G. Smith. „Hybrid Quantum-Classical Computing Architectures“. In: *Proceedings of the 3rd International Workshop on Post-Moore Era Supercomputing, 2018*. 2018. URL: http://par.nsf.gov/biblio/10084839 (zitiert auf S. 26).

[Sho94] P. W. Shor. „Algorithms for quantum computation: discrete logarithms and factoring“. In: *Proceedings 35th Annual Symposium on Foundations of Computer Science*. Nov. 1994, S. 124–134. DOI: 10.1109/SFCS.1994.365700 (zitiert auf S. 27, 28, 43–47).

[Ske81] D. Skeen. „Nonblocking Commit Protocols“. In: *Proceedings of the 1981 ACM SIGMOD International Conference on Management of Data*. SIGMOD '81. Ann Arbor, Michigan: ACM, 1981, S. 133–142. ISBN: 0-89791-040-0. DOI: 10.1145/582318.582339. URL: http://doi.acm.org/10.1145/582318.582339 (zitiert auf S. 34, 38, 41, 59, 63, 65, 66, 68, 69).

[ST17] M. van Steen, A. Tanenbaum. *Distributed Systems, 3rd ed.* distributed-systems.net, 2017 (zitiert auf S. 28, 29, 31, 34–41, 49, 52, 59–65, 67, 68).

[VA15] R. Van Renesse, D. Altinbuken. „Paxos Made Moderately Complex“. In: *ACM Comput. Surv.* 47.3 (2015), S. 42–1. DOI: 10.1145/2673577 (zitiert auf S. 13, 14, 28–31, 49, 50, 53).

[Ved10] V. Vedral. „The Elusive Source of Quantum Speedup“. In: *Foundations of Physics* 40.8 (März 2010), S. 1141–1154. ISSN: 1572-9516. DOI: 10.1007/s10701-010-9452-0. URL: http://dx.doi.org/10.1007/s10701-010-9452-0 (zitiert auf S. 47, 69).

[Vid03] G. Vidal. „Efficient Classical Simulation of Slightly Entangled Quantum Computations“. In: *Physical Review Letters* 91.14 (Okt. 2003). ISSN: 1079-7114. DOI: 10.1103/physrevlett.91.147902. URL: http://dx.doi.org/10.1103/PhysRevLett.91.147902 (zitiert auf S. 46, 47).

[WZ82] W. K. Wootters, W. H. Zurek. „A single quantum cannot be cloned“. In: *Nature* 299.5886 (1982), S. 802. DOI: 10.1038/299802a0. URL: https://doi.org/10.1038/299802a0 (zitiert auf S. 21).

[Yan07] N. S. Yanofsky. „An Introduction to Quantum Computing“. In: (2007). arXiv: 0708.0261 [quant-ph] (zitiert auf S. 24).

[Zha07] W. Zhao. „A Byzantine Fault Tolerant Distributed Commit Protocol“. In: *Third IEEE International Symposium on Dependable, Autonomic and Secure Computing (DASC 2007)*. Sep. 2007, S. 37–46. DOI: 10.1109/DASC.2007.10 (zitiert auf S. 35, 59).

[Zur06] W. H. Zurek. „Decoherence and the Transition from Quantum to Classical — Revisited“. In: *Quantum Decoherence* (2006), S. 1–31. DOI: 10.1007/978-3-7643-7808-0_1. URL: http://dx.doi.org/10.1007/978-3-7643-7808-0_1 (zitiert auf S. 26).

Alle URLs wurden zuletzt am 5. 11. 2019 geprüft.

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Ort, Datum, Unterschrift