Quantum Data Centres in the Presence of Noise

Kenneth Campbell

School of Electrical and Electronic Engineering

University of Leeds

Submitted in accordance with the requirements for the degree of

Doctor of Philosophy

December 2025

I confirm that the work submitted is my own, except where work which has formed part of jointly authored publications has been included. My contribution and the other authors to this work has been explicitly indicated below. I confirm that appropriate credit has been given within the thesis where reference has been made to the work of others.

The work described in Chapter 3 will soon be submitted to the journal SoftwareX and arXiv as a paper entitled "dqc_simulator: an easy-to-use distributed quantum computing simulator", by Kenneth Campbell. However, the contents of that manuscript deviate greatly from Chapter 3.

Chapters 4 and 5 of this thesis are based on the paper "Quantum data centres: a simulation-based comparative noise analysis" by Kenneth Campbell, Ahmed Lawey, and Mohsen Razavi, which was published in IOP Quantum Science and Technology on the 23rd of December 2024. I, Kenneth Campbell, produced all calculations, results, the simulator used to obtain those results, and all plots and images. I also wrote all drafts of the publication. Ahmed Lawey and Mohsen Razavi supervised my work and provided editorial and technical comments.

Chapter 6 is based upon the paper "Combatting noise in near-term quantum data centres" by Kenneth Campbell, Ahmed Lawey and Mohsen Razavi, which is available at arXiv:2601.14845 and has been submitted for publication. As before, I developed all code and calculations, and produced the results, plots and images, in addition to all publication drafts. Mohsen Razavi and Ahmed Lawey supervised my work and provided editorial and technical comments.

Acknowledgements

A PhD thesis is rarely a truly solitary endeavour and this one is no exception. I am hugely grateful for all of the help and support that I have received during my time at the University of Leeds.

I would like to acknowledge and thank Mohsen Razavi and Ahmed Lawey for their supervision and support during my PhD. Their advice has informed the way that I think about problem solving and research, and is something that I will carry through into the rest of my career and life. I also sincerely thank them for their understanding, compassion and support throughout my PhD and especially during some of the challenges in my personal life that arose during my PhD.

I would also like to thank my colleagues Javier Rey Domínguez, Masoud Ghalaii, Zhaohui Liu and Sahana Dermal for many useful and interesting discussions.

Above all I would like to thank my family and friends without whose love and support none of this would have been possible. I would especially like to thank my girlfriend, Sophie Higham, for the love, humour and whimsy she brings to my life and her patience during the intrinsic tumultuousness of a PhD; my parents, Sian and Ewan Campbell, for the unwavering love and support they have always shown me and the space that they have provided for me to follow my inquisitiveness and interests; my brother Finlay for his wit and warmth and for expanding my world with his intelligence and insight; my Taekwon-do family for creating a home several hours from where I grew up; and Jay Kerslake for helping me see the world in a new light and always being there for me in the difficult times and the good ones. I love you all.

Abstract

Distributed quantum computing is a promising way of overcoming the scalability challenges of quantum computers, but establishing quantum links over large distances remains difficult. Therefore, in this thesis, we explore quantum data centres (QDCs), in which multiple quantum processing units (QPUs) are housed together within the same warehouse and linked together over short links. By considering such a setting, we avoid the challenges of long-distance quantum communications while retaining the scalability of distributed architectures.

We focus primarily on understanding and quantifying the role of noise in QDCs. We compare the impact of different types of noise in the QDC setting and explore various types of inter-QPU gate. Both individual inter-QPU gates and an assortment of larger distributed quantum circuits are investigated. Based on our findings, we evaluate strategies for the handling of noise, considering both quantum error detection and entanglement distillation based solutions. We also provide a preliminary investigation into the use of QDCs for generating entangled resource states suitable for quantum communications or measurement based quantum computing applications.

A major contribution of this thesis is the development of a novel simulation framework for distributed quantum computing, which formed the basis of many of the results obtained. The framework enables users to easily specify distributed quantum circuits or import circuits designed for a single-QPU and convert them to a distributed circuit. Many of the complications of interpreting a distributed quantum circuit, such as the management of communication qubits are also handled automatically. Moreover, the framework's built-in interpreter can interpret

common building blocks of distributed quantum computing, such as inter-QPU gates.

Abbreviations

Acronym	Meaning
QC	Quantum Computer
QPU	Quantum Processing Unit
QDC	Quantum Data Centre
CNOT	Controlled-NOT
CU	Controlled-Unitary
EPR	Einstein-Podolsky-Rosen
QED	Quantum Error Detection
BSM	Bell state Measurement
SDK	Software Development Kit
UML	Unified Modelling Language
OOP	Object Oriented Programming
NISQ	Noisy Intermediate-Scale Quantum
MBQC	Measurement-Based Quantum Computing
AE	Amplitude Estimation
DJ	Deutsch-Jozsa
GHZ	Greenberger-Horne-Zeilinger
QAOA	Quantum Approximation Optimisation Algorithm
QFT	Quantum Fourier Transformation
QPE	Quantum Phase Estimation
VQE	Variational Quantum Eigensolver

Table 1: Common acronyms

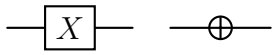
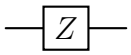
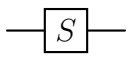
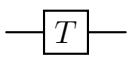
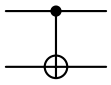
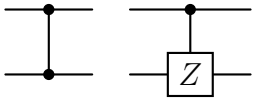
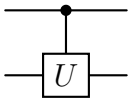
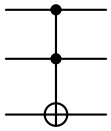
Name	Symbol	Matrix
Pauli-x, X or σ_x		$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$
Pauli-y, Y, or σ_y		$\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$
Pauli-z, Z, or σ_z		$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$
Hadamard		$\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$
Phase or S		$\begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$
$\frac{\pi}{8}$ or T		$\begin{pmatrix} 1 & 0 \\ 0 & e^{\frac{i\pi}{4}} \end{pmatrix}$
CNOT, CX, or controlled-NOT		$\mathbb{1}_2 \oplus \sigma_x$
CZ or controlled-Z		$\mathbb{1}_2 \oplus \sigma_z$
CU or controlled-unitary		$\mathbb{1}_2 \oplus U$
SWAP		$\mathbb{1}_1 \oplus \sigma_x \oplus \mathbb{1}_1$
Toffolli, CCNOT, or CCX		$\mathbb{1}_6 \oplus \sigma_x$
Entangling gate		N/A

Table 2: Some common gates and their representations. $\mathbb{1}_n$ refers to the $n \times n$ identity matrix. $\oplus$ is the direct sum, when appearing in the final column, and should not be confused with the circuit notation for the Pauli-x gate.

Contents

List of Figures

LIST OF FIGURES

Chapter 1

Introduction

1.1 Introduction

Few areas of physics have garnered as much attention in recent years as quantum computing. Quantum computing offers the chance to revolutionise the fields of chemistry, medicine, economics, cyber-security, meteorology, and many others, through its ability to simulate otherwise inimitable quantum systems [9] and solve problems believed to be difficult for classical computers [10, 11]. However, despite the significant promise of the field and its relative longevity [12–15], there is yet to be a quantum computer able to tackle useful problems that a classical computer cannot. A key reason for this is the immense difficulty in fabricating a single, monolithic quantum processor with sufficient high-quality, individually addressable qubits [16, 17] to perform the algorithms that yield the most quantum advantage. As well as the many modality-specific engineering challenges to scaling [16, 17], there is a fundamental challenge in keeping qubits separate and individually addressable while also enabling them to strongly interact in a controlled fashion, when multi-qubit gates are required. This issue becomes harder and harder to surmount when more and more qubits are kept in proximity to one another. One way around these engineering challenges is to connect the smaller quantum processors, that can already be successfully manufactured, via a quantum network, to form a single distributed machine.

Distributed computing is not a new idea and is well established in modern classical computing [18]. However, complications arise when it is attempted to

convert this idea to the quantum regime. For example, the no-cloning and no-broadcasting theorems [19, 20] prohibit the amplification of unknown signals by imposing that the state of an arbitrary, unknown qubit cannot be copied. With this limitation, it is far harder than in classical systems to get around the inevitable losses that occur when information is transmitted over distance. Quantum gates between different processors are also more computationally expensive than those conducted on monolithic processors [21], due to the high noise and latency of entanglement creation and distribution, which are both key requirements of non-local (remote) quantum gates.

The first of these problems, loss, can be overcome using quantum repeaters [22], but this technology is far from maturation [23], and so distribution of quantum computing resources over large distances is unlikely in the immediate future. This does not however prohibit the, rather less explored, idea of linking quantum processors over short distances to create a single, more powerful device. We consider such scenarios in which several quantum processors are housed in the same warehouse facility and linked together, which we call a quantum data centre (QDC). QDCs mitigate the issues with loss that plague larger quantum networks, but, at least for matter-based QDCs, which we focus on, typically distribute entanglement using the same transduction between static qubits and flying photonic qubits that are needed in larger networks, enabling QDCs to potentially be integrated into a larger quantum internet in the future. However, the second issue, inter-QPU noise, remains an issue.

Prior to commencement of this work, it remained to be seen how a medium-sized QDC would perform, when realistic errors and architectural limitations are accounted for. It also remained to be seen whether such a machine is feasible at all, and how it would compare to the monolithic case. QDCs enable arbitrary scalability of the number of qubits to be achieved while retaining the quality of the qubits and of any gates conducting on a given QPU. However, this is achieved at the cost of imposing an additional source of noise, inter-QPU entanglement error, that is not found in monolithic machines. The fundamental question underlying all of the work in this thesis is whether that is too high a price to pay.

In the following sections, we provide broader context for where this project fits in. To this end, we give a brief overview of quantum computing in Sec. 1.2 and of

distributed quantum computing in Sec. 1.3. After that, the main contributions of the thesis are highlighted in Sec. 1.4 and an outline of the thesis as a whole is given in Sec. 1.5.

1.2 Quantum computing: an overview

Quantum computing was first proposed independently by Paul Benioff [24] and Yuri Manin [25] and popularised by Richard Feynman [26], who also formalised many of the concepts. Feynman and Manin motivate the importance of quantum computing with the argument that simulating a quantum system on a classical computer incurs an exponential increase in time cost as the size of the system increases. As quantum physics represents our most accurate depiction of reality to date, this means that many features of the real world cannot be captured with a classical computer alone.

Shortly afterwards, the question arose of whether quantum computers may also offer an advantage for certain classical problems and several somewhat artificial algorithms were contrived [27–29] that showed such an advantage. However, it was not until Shor developed an algorithm that factored numbers exponentially faster than any known classical algorithm [10] that the excitement really grew around the field of quantum computing.

Despite this excitement, doubts remained about whether quantum computers would ever be realised, due to their sensitivity to errors. These doubts were largely dispelled by the invention of the first quantum error correction scheme [30] and the proof that uncorrelated quantum errors can be discretised, which means that the ability to correct for a finite set of errors is sufficient to correct any uncorrelated error [31].

Alongside these algorithmic advances, the building blocks of quantum computers began to be realised experimentally [32, 33]. These experimental advances continued, ushering in the era of noisy intermediate-scale quantum (NISQ) computers [34]. However, NISQ devices lack the resources to correct all errors in the system while running large quantum algorithms and so larger more powerful devices are needed. To implement fault-tolerant devices, for which error correction fixes more errors than it causes, a greater number of the fundamental information

carriers in quantum computing, qubits, are needed and the probabilities of errors occurring on those qubits must be low.

In this thesis, we focus on QDCs as a way of scaling up quantum computers towards the fault-tolerant regime. In particular, we consider proof of principle demonstrations of distributed quantum computing as a stepping stone to bridge the gap between current, single-processor NISQ devices and larger, fault-tolerant devices.

1.3 Distributed quantum computing: an overview

The idea of distributed quantum computing is far from novel. Around the same time that Shor's factoring algorithm was developed, Lov Grover and, independently, Richard Cleve and Harry Buhrman first proposed that quantum computing could be distributed using non-local quantum communication [35, 36].

These ideas gained traction when Cirac et al. found that distributed quantum computing could perform the quantum phase estimation algorithm well, even in the presence of noise [37]. Soon afterwards, Eisert et al. and Collins et al. gave theoretical proposals and resource optimisation for the efficient implementation of remote gates [38, 39].

With the theoretical foundations laid, more explicit, hardware-specific suggestions for how distributed quantum computing could be experimentally implemented were made [40–43]. Around the same time, distributed versions of seminal quantum computing algorithms were formulated [3, 44].

Some actual, albeit small-scale, experimental implementations of teleportation-based gates have also been carried out [45–50]. However, these schemes were either probabilistic [45, 46, 49, 50] or were conducted on monolithic [47] or multi-core [48] devices, in which different quantum processing units (QPUs) share a control system or housing [51]. This latter type of device differs from the QDCs that we focus on, in that the distances between connections are typically shorter, with quantum processing cores being confined to the same vacuum chamber, rack, refrigerator or optical table [51]. By contrast, in QDCs, distinct QPUs, each

housed in their own vacuum, refrigerator or other similar container are connected via a local area network. This distinction means that QDCs are more likely to require a photonic connection between QPUs that more closely resemble those used in a larger metropolitan-scale or larger device and the engineering challenges needed to create larger vacuum chambers or refrigerators are avoided. Until very recently, no deterministic realisation of the QDC paradigm that we focus on had been experimentally implemented. This changed in early 2025 with the work done in Ref. [52] in which a remote gate and a very small, two qubit-Grover's algorithm were conducted on two-qubit trapped ion QPUs separated by a 2m photonic link.

Ref. [52] did not distinguish between the impact of different error types and prior to this thesis there was a dearth of theoretical work doing so either. Moreover, Ref. [52] considered only two processing qubits, which actually participate in quantum algorithms, with the remainder of the qubits being set aside for quantum communication. Therefore, there is a space in the literature for better understanding the small to medium-sized QDCs likely implementable in the near-term. It is this vacuum that we seek to fill with this thesis.

1.4 Main contributions of this thesis

Underlying many of the results in this thesis is the dqc_simulator package, which we developed. This novel distributed quantum computing simulator dramatically simplifies numero-analytical investigation of distributed quantum computing and has use cases that extend beyond the scope of this work. We introduce and discuss this simulator in Chapter 3.

In Chapter 4, we demonstrate the limitations of first-order error analysis of individual remote gates in the QDC context, relative to full classical simulation. We derive analytical expressions for the output fidelity of various types of remote gate in the presence of entanglement noise only, assuming that the input state is separable. We then use classical simulation to ascertain the relative performance of various types of remote gate in the presence of different types of noise. The relative impact of noise in the inter-QPU entanglement, local two-qubit gates and memory depolarisation is determined for the various types of remote gate considered.

Chapter 5 extends our analysis to larger quantum circuits containing many remote gates, giving errors a greater chance to propagate. In Chapter 5, we determine the relative impact of the aforementioned noise types on these larger circuits. We investigate the propagation of errors as the number of qubits in the quantum circuits and hardware explored are increased. We also make several observations on the feasibility of near-term QDCs, constructed using matter-based qubits with photonic interconnects.

The results in Chapters 4 and 5 are published in IOP Quantum Science and Technology [53] and are also available on the arXiv at arXiv:2407.10769.

Chapter 6 addresses many of the challenges to QDC implementation raised in Chapters 4 and 5. The notion of applying quantum error detection in a localised fashion to remote gates is proposed and explored, and conventional entanglement distillation techniques are considered. We compare the performance of both approaches to QDC error handling with a view to near term demonstrations of QDC operation.

The results in Chapter 6 are available on the arXiv at arXiv:2601.14845 and have been submitted for publication.

As part of our discussion of future outlook, we also present some preliminary results for which further exploration is desirable. In particular, we explore the use of noisy QDCs to create generalised GHZ states [2, 54] and cluster states [55] using increasing numbers of QPUs.

1.5 Thesis outline

The remainder of this thesis is organised as follows. In Chapter 2, we discuss the technical background information needed to understand the remainder of the thesis. Chapter 3 introduces and discusses the simulation tool underlying much of the work in this thesis. Chapter 4 follows on from this with a detailed exploration of noise in remote gates, which are the fundamental building blocks and typical bottlenecks of distributed quantum circuits. This analysis is expanded to include larger distributed quantum circuits with multiple remote gates in Chapter 5. Having ascertained the most damaging type of noise in Chapters 4 and 5, Chapter 6 goes on to investigate methods for combatting such noise. Finally,

we summarise and conclude our discussion in Chapter 7 and discuss some less developed work and potential avenues for future research. Further details on the analytical results from Chapter 4 are provided in Appendix A. The impact of input state on the results in Chapters 4 and 5 is discussed in Appendix B. Appendix C provides analytical calculations to support the the discussion in Chapter 6. Finally, Appendix D explores the reasons behind the limited impact of one of the quantum error detection schemes that we consider in Chapter 6.

Chapter 2

Theoretical background

Distributed quantum computing represents the intersection of some of the most elegant and significant results in all of quantum mechanics. Therefore, it is necessary to understand these ideas before comprehension of the field can be achieved. A full discussion of most of the necessary concepts can be found in Nielsen and Chuang's textbook [56], but we include discussion of some of the most essential ideas here for convenience.

2.1 The basics of quantum computing

2.1.1 Superposition and qubits

Perhaps the most central concept to quantum computing in general is the idea of quantum superposition.

In a traditional, classical computer, information is encoded as a bit, which can be thought of as nothing more than a simple switch. A bit can exist in one of two absolute states, 'off' or 'on', which can be renamed 0 or 1, respectively, for convenience.

In the quantum world, things become somewhat fuzzier. Somewhat similarly to before, quantum states can be encoded in a binary basis, which in Dirac's braket notation includes basis eigenstates: $|0\rangle$ and $|1\rangle$. Tellingly, this basis is often referred to as the computational basis. However, quantum states need not be absolute but can instead exist as superpositions of the basis eigenstates: our

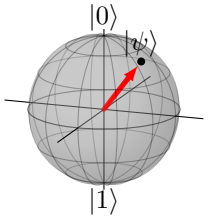


Figure 2.1: The Bloch sphere. The red arrow indicates a pure state vector of the form given by Eq. (2.1).

switch can exist as a linear combination of 'off' and 'on'. Such a state can be written down as:

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

where the vector, $|\psi\rangle$, encoding the quantum state is referred to as a wavefunction, and, to ensure our wavefunction remains physically meaningful, the coefficients α, $\beta \in \mathbb{C}$ are normalised, such that $|\alpha|^2 + |\beta|^2 = 1$.

In concession to this distinctly non-classical behaviour, the most commonly used fundamental information carriers in quantum computing are referred to as quantum bits, or qubits, and most generally exist in the state described by Eq. (2.1). It is also frequently useful to represent the states of qubits as matrices, to more conveniently utilise the linearity of quantum mechanics. By convention, the computational basis states from (2.1) are typically mapped as:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \tag{2.2}$$

and

$$|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}. \tag{2.3}$$

These basis states define a two-dimensional Hilbert space, a type of finite-dimensional complex vector space that frequently arises in quantum mechanics [56], and have a convenient geometric representation as opposite poles on a sphere, which is typically referred to as the Bloch sphere. A depiction of the Bloch sphere can be seen in Fig. 2.1.

Any single-qubit state that has the form given in Eq. (2.1) can be thought of as a vector touching the surface of the Bloch sphere. Such states are referred to as *pure* states.

2.1.2 Noise and fidelity

In reality, a truly closed system can never be obtained and additional noise is imposed on the system by its environment. This makes the pure quantum states discussed in the previous section an idealisation.

Realistic states will instead be, to some extent, a statistical mixture of pure quantum states. Such a mixture is fundamentally different from the quantum superposition discussed previously in that it is valid to think of the system as existing in one of several pure states with a given probability. The state of the system is absolute but unknown to us. This is not true of a pure quantum superposition state, which should be thought of as actually in the superposition state. For $\alpha, \beta > 0$, it is not accurate to say that a qubit in state $|\psi\rangle$ from Eq. (2.1) is in state $|0\rangle$ or state $|1\rangle$ and that measurement simply reveals this [57–59]. Measurement is fundamentally altering the state itself - instantaneously collapsing the superposition to one of its constituent basis states.

This intrinsic difference means that pure states and statistical mixtures need to be represented differently. To this end, statistical mixtures are normally described using the density matrix formalism.

A density matrix ρ has the form

$$\rho = \sum_i p_i \left|\psi_i\right\rangle \left\langle\psi_i\right|, \tag{2.4}$$

where $|\psi_i\rangle$ are pure quantum states and p_i are the probability of the system being in the corresponding pure state. If the state represented by ρ is pure, then the sum has just one term with probability one.

Within this formalism, it is easier to quantify how much a noisy state deviates from the one ideally expected. The most common way to achieve this is through the use of fidelity. Considering a desired state, ρ_{ideal}, and the actual state produced, ρ_{noisy}, fidelity can be defined as [60]

$$F(\rho_{\text{ideal}}, \rho_{\text{noisy}}) = \left(\text{Tr}\sqrt{(\rho_{\text{ideal}})^{\frac{1}{2}}\rho_{\text{noisy}}(\rho_{\text{ideal}})^{\frac{1}{2}}}\right)^2, \tag{2.5}$$

where Tr is the trace operator.

The fidelity can be visualised more vaguely as a distance measure indicating deviation from ideality. A state with a fidelity of one is 100% identical to the

ideal state we were attempting to produce. By contrast, a fidelity of zero implies that nothing of the ideal state remains in the output.

In the specific case that the ideal state was a pure state, $|\psi\rangle$, (2.5) simplifies to

$$F(|\psi\rangle\langle\psi|, \rho_{\text{noisy}}) = \langle\psi|\rho_{\text{noisy}}|\psi\rangle . \tag{2.6}$$

It is also possible to define fidelity as the square root of Eqs. (2.5) and (2.6) [56]. Nonetheless, in my project, I adopt the definitions (2.5) and (2.6), unless otherwise stated.

2.2 The advantage of quantum computers

The concept of quantum superposition introduced in section 2.1.1 provides the first clues as to how quantum physics can provide an advantage for computing purposes. The superposition state, shown in Eq. (2.1), has more degrees of freedom than classical binary states, 0 or 1, and allows us to encode information in Hilbert spaces that grow exponentially quickly with the number of qubits. This affords quantum computers a theoretically exponential advantage in information storage for a given number of qubits relative to the same number of bits.

Unfortunately, as discussed previously, measurement instantaneously collapses superposition states into one of the constituent basis states, meaning that results of any computation have to be read out as bits. At first glance, it seems that this destroys any advantage a quantum computer may have, but it is actually still possible to make use of this exponentially larger information density during the computation itself. Quantum entities have a dual nature, and can exhibit the properties of both particles and waves [61]. As such, it is possible to interfere quantum states during computation to enhance the probability of measuring the correct result in much the same way as the amplitude of a classical signal can be increased via constructive interference. All of this means that the many possible computational outcomes that can be encoded on the states of a multi-qubit register can be considered at once, with the outcome corresponding to the problem solution being preferentially selected prior to the loss of information during measurement.

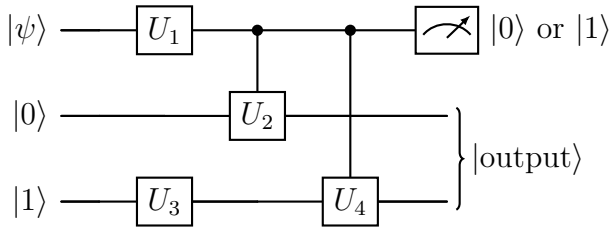


Figure 2.2: Example quantum circuit. U_1, U_2, U_3, and U_4 are unitary operations.

2.3 Quantum gates and models of quantum computation

Qubits alone are not enough to form a quantum computer, we must also be able to manipulate those qubits, so as to actually perform computations on them.

There are several ways of viewing these manipulations. The first and most commonly used is to break down any manipulation of qubits into quantum gates, in analogy to classical logic gates. Algorithms are depicted via diagrams like the one shown in Fig. 2.2 and the diagrammatic symbols for various operations are shown in Table 2.

Assuming a closed system, the quantum gates can mathematically be described as unitary operations, which act on a 'ket' state of the form depicted in Eq. (2.1) as $U|\psi\rangle$ and on its complex conjugate 'bra' state as $\langle\psi|U^\dagger$. In the density matrix formalism, gates are applied according to the expression $U\rho U^\dagger$. The unitarity of the operations means that they are reversible, unlike many gates in classical circuits.

For further convenience, any unitary quantum gate can be efficiently approximated to arbitrary precision using a universal set of quantum gates [56], allowing universal computation to be conveniently achieved if the gates in this set can be implemented on the hardware. Several universal sets exist, but perhaps the most commonly used universal set consists of the Hadamard, phase, controlled-NOT (CNOT), and T, or $\frac{\pi}{8}$, gates [56]. The corresponding circuit symbols and matrix representations are shown in Table 2, along with those for a few other commonly used gates.

2.4 Quantum communication and quantum networks

Up to this point, quantum computing has been discussed at length, but, for a computer to be distributed, communication is also important. As the main goal of this project is to increase the number of high-quality qubits that can be brought to bear on a given problem, the communication considered needs to be quantum in nature.

The key to quantum communications is quantum entanglement, but, to understand this idea, it is necessary to think about how we represent the state of multiple qubits.

The most basic way of representing the states of multiple independent qubits is to combine the state of each qubit using the tensor product, through which we can represent the state of two qubits, with states $|\psi_1\rangle_{q_1}$ and $|\psi_2\rangle_{q_2}$, respectively, as

$$|\psi_1\rangle_{q_1} \otimes |\psi_2\rangle_{q_2} \,. \tag{2.7}$$

Equation (2.7) is often shortened to

$$|\psi_1\psi_2\rangle_{q_1,\,q_2} \equiv |\psi_1\psi_2\rangle \,, \tag{2.8}$$

for convenience, provided that the order in which the states appear is clear.

In the density matrix representation, we can instead write

$$\rho_{1,\,2} = \rho_1 \otimes \rho_2 \equiv \rho_1\rho_2, \tag{2.9}$$

where ρ_1 and ρ_2 are the states of q_1 and q_2, respectively.

States that can be written in the form of Eqs. (2.7) to (2.9) are referred to as separable states.

The picture changes somewhat when quantum correlations between states are present. Consider, for example, the state

$$\left|\Phi^+\right\rangle = \frac{1}{\sqrt{2}}\left(|00\rangle_{q_1,\,q_2} + |11\rangle_{q_1,\,q_2}\right), \tag{2.10}$$

where all symbols are as previously defined. This is a valid quantum state, written in terms of normalised basis states, but cannot be written in the separable form of Eq. (2.7) or Eq. (2.9). When this is true, a state is said to be entangled.

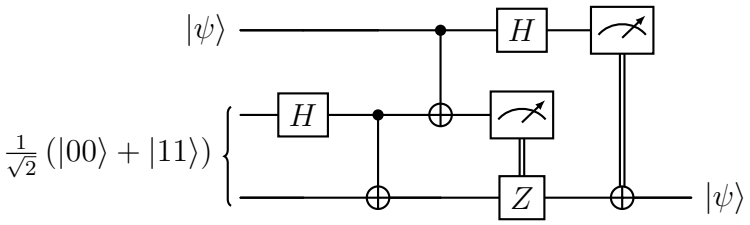


Figure 2.3: Teleportation circuit. The double lines indicate that the target gates are applied when the classical measurement results are 1.

Entangled states possess deep correlations between their constituent qubits. For a maximally entangled bipartite state, such as the one in Eq. (2.10), measurement on one qubit completely defines what the result of a measurement, in the same basis, on the other qubit must be. For qubits q_1 and q_2, collectively in state $|\Phi^+\rangle$ from (2.10), a measurement made on q_1, which could be billions of miles away from q_2, will *instantaneously* collapse the state of q_2 into one of its eigenstates, without the need for q_1 and q_2 to be fixed in a potentially unknown eigenstate beforehand [57–59, 62]. This is despite the fact that whichever of Alice and Bob makes their measurement first will obtain truly random results when viewed independently of their partner's measurement. All of this holds true in all basis states, provided that the measurement on both qubits is made in the same basis. The same ideas can also be generalised to states with more than two qubits.

Maximally-entangled states are so important to quantum communications that examples of them are often given special names and labels. For example, the designation of the state defined by Eq. (2.10) as $|\Phi^+\rangle$ has been widely adopted in quantum physics literature. $|\Phi^+\rangle$ is one of the widely utilised Bell states, which are elements of the set $\{|\Phi^\pm\rangle = \frac{1}{\sqrt{2}}(|00\rangle \pm |11\rangle), |\Psi^\pm\rangle = \frac{1}{\sqrt{2}}(|01\rangle \pm |10\rangle)\}$. The Bell states are also commonly referred to as Einstein-Podolsky-Rosen (EPR) pairs [56].

As a taste of the power of quantum entanglement, consider the circuit shown in Fig. 2.3, in which the state described by Eq. (2.10) has been distributed between the middle and bottom qubits. Working through the algebra reveals that, at the end of the circuit, the bottom qubit is left in the arbitrary, initial state of the top qubit. We have teleported the quantum state from the top to the bottom qubit!

Crucially, by distributing entanglement between different locations, quantum states can be moved between those locations. As entanglement is a non-local phenomenon, quantum teleportation can, in principle, be achieved over any distance.

Unfortunately, as has been a common theme of the quantum technologies discussed, what is achievable theoretically is often more challenging to realise practically. Typically, the qubits used to implement quantum computers are static and fixed to a single location, meaning that an intermediary 'flying' qubit, which will almost certainly be a photon, is required to distribute entanglement between geographically separated locations A and B. Generating a flying qubit entangled with a static one is non-trivial and so, for the distributed quantum computing context, it is helpful to split static qubits into two types: processing qubits, upon which quantum programs can be run, and communication qubits, which are able to absorb and emit photons in such a way that they are entangled with the communication qubit. The communication and processing qubits often need to be physically different to avoid crosstalk or exciting photon emission from multiple qubits at once [63]. For example, when using atomic or ionic distributed quantum computers, the communication qubits may be of a different atomic species to the other qubits on a given quantum processing unit (QPU) node and would often be in an optical resonator.

There are three broad ways in which entanglement can be distributed: probabilistically, without heralding; probabilistically, with heralding; and deterministically. In the unheralded probabilistic case, it is not certain if an entangled pair will be distributed between the nodes in any one run of the scheme, and it is typically only known afterwards whether entanglement was present, through the use of post-selection, once the entire experiment, computation or process is complete. In a heralded entanglement distribution, there is again no a priori guarantee, even in the ideal case, that entanglement will be distributed between the nodes during a given run of the entanglement distribution scheme, but, through measurement, the success or failure of entanglement distribution is revealed during the experiment. This means that the scheme can be repeated until it is known entanglement has been generated. Finally, there are deterministic schemes in which entanglement is (ideally) always generated each time the scheme is implemented.

As soon as any realistic imperfections are considered, even some deterministic schemes may need to be repeated. Losses will occur in the system, due to failed coupling between the flying and communication qubits, lack of detection during measurement, and channel losses as the flying qubit (photon) travels through the space between nodes. The higher the losses are, the more times each scheme must be repeated and the greater the latency introduced. In many cases, this can be a very signficant bottleneck in performance [22].

Losses are far from the only imperfection plaguing quantum communications. Imperfect generation and measurement of entangled pairs of qubits, collectively referred to as ebits, along with environmental noise, reduce the fidelity of the ebits, regardless of how many times entanglement distribution is repeated, and introduce impurity to the system. Instead of the pure states obtained by ideal entanglement distribution, in practice, ebits may be statistical mixtures of the intended pure state and unintended erroneous states.

To circumvent such imperfections, a process called entanglement distillation can be used [7, 8, 64, 65]. The idea of entanglement distillation is that additional, redundant ebits are generated between nodes and then entangled with one another via local multi-qubit operations on each node. The quantum correlations between redundant and original qubits mean that measuring the redundant qubits allows errors to be detected in the entangled pair of interest. Depending on the scheme, detected errors mean that either the whole process is abandoned and repeated until errors are not detected or that those errors are corrected. For the former type of scheme, an arbitrarily high fidelity can be obtained, provided that the initial fidelity of the ebits is greater than 50% and enough time is expended to repeat the scheme until success. For the latter type of scheme, there is normally a theoretical limit to the fidelity obtainable and the fidelity of the original, undistilled pairs must be quite high for the schemes to work. In reality, both types of scheme are limited by the error rate of the local gates used during the distillation process.

If entanglement is to be generated over larger distances, a third step is needed to overcome the losses incurred and obtain a non-vanishing rate of entanglement. This step is usually referred to as entanglement swapping. Entanglement swapping can be understood by considering three spatially-separated nodes, Alice, Bob, and Charlie, which each contain qubits (see Fig. 2.4). For simplicity, let Alice and

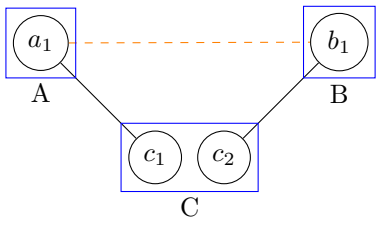


Figure 2.4: Illustration of entanglement swapping principle. Nodes Alice, Bob, and Charlie, labelled A, B, and C, respectively, are at geographically separated locations. If qubit a_1 is entangled with c_1 and c_1 with c_2 (entanglement shown as solid line), then a Bell state measurement on Charlie's qubits will generate entanglement between Alice and Bob's qubits (dashed line).

Bob each have one qubit and Charlie have two. If Alice and Bob's qubits are both entangled with a different one of Charlie's, then a measurement of Charlie's qubits in the Bell basis (normally called a Bell state measurement or BSM), will entangle Alice and Bob's qubits with one another. Recursive use of this phenomenon could in theory allow entanglement to be distributed over arbitrary distances, but entanglement swapping can degrade the fidelity [22]. As such entanglement swapping necessitates more entanglement distillation, and consequently comes with a time and resource cost.

Due to these complications, the practical distribution of entangled states over arbitrarily large distances at a useful rate remains an open implementation problem, and the devices needed for this, quantum repeaters, are unlikely to be practical for at least another 10 to 15 years. With a view to near-term demonstration of scalable distributed quantum computer, we therefore focus on short distances on the order of metres or less. In such settings, losses are significantly reduced, and the need for entanglement swapping is much less pressing.

2.5 Quantum error correction and detection

In a similar vein to the entanglement distillation introduced in the previous section, redundancy and quantum entanglement can be used to correct local errors too.

To understand how quantum error correction works, it is initially helpful to consider classical error correction. Perhaps the simplest scheme to conceive of is

a classical three-bit repetition code. In a classical repetition code, the value of a single bit is copied into multiple other bits. For example, a bit with the value 0 could be represented as the logical bit $0_\mathrm{L} = 000$. In a similar vein, $1_\mathrm{L} = 111$. Then, if a single bit-flip error occurs, such that 000 goes to 010 for example, we can simply measure all three bits and take a majority vote to see what the logical state should be. We can then simply flip the incorrect bit and proceed with the computation.

Even from this simple classical example, several issues immediately become apparent. Firstly, due to the redundant encoding, there is a resource cost to such error correction. Where before one bit sufficed, we now need several. Furthermore, if we were to have two bit-flip errors, then the majority vote system would correct to the wrong logical state. As such, the code is only useful if two bit errors are sufficiently unlikely. These features generalise to all error correcting codes, including quantum ones. For quantum error correction we need many physical qubits to provide the necessary redundancy, but errors must be sufficiently unlikely for the code to work, necessitating high-quality qubits and gates.

To conveniently indicate these metrics it is common to refer to the $[[n, k, d]]$ properties of a code, where n is the number of physical qubits required to encode k logical qubits, and d is the code distance. The code distance is the number of errors required to go from one logical basis state to another. In the classical three-bit repetition code example given earlier, three bit flips are needed to go from 0_L to 1_L and so $d = 3$. Another common metric used is the threshold for a given code, which is the maximum error probability that can be tolerated before there is no benefit to using the code. In more complicated scenarios, the threshold also depends on the decoder used to identify what errors occur where, and so there can be more than one threshold for a given code.

There are also additional complications when going from classical to quantum error correction. As discussed previously, we cannot make copies of arbitrary quantum states and direct measurement of qubits collapses superposition states. Fortunately, quantum entanglement is again useful here. We can represent an arbitary state in the three-qubit quantum repetition code as the entangled state:

$$|\psi\rangle_\mathrm{L} = \alpha\,|0\rangle_\mathrm{L} + \beta\,|1\rangle_\mathrm{L} = \alpha\,|000\rangle + \beta\,|111\rangle\,. \tag{2.11}$$

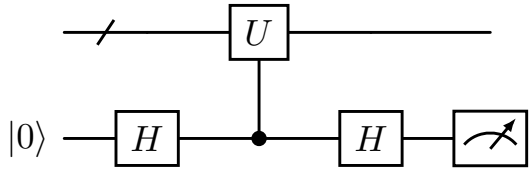


Figure 2.5: Circuit used to measure an arbitrary unitary operator U.

Rather than measure all of the qubits and take a majority vote, which would collapse the superposition, we can measure the parity of subsets of the qubits and piece this information together to infer where errors have occurred. This does not give us information about the quantum state we are in and so does not destroy the superposition but does tell us if we have left the space of allowed logical states, allowing us to infer if errors have occurred. Such measurements are known as stabiliser measurements.

To achieve this, we use unitary operators that have eigenvalue +1 when acted on the basis state and measure them using a circuit of the form shown in Fig. 2.5, which is based on a figure in Ref. [56], and requires an additional ancilla qubit on top of the qubits used to encode the logical state. The idea is that errors, which can be modelled as operations E, will anti-commute with the stabilisers to give an eigenvalue of -1, corresponding to a measurement value of 1 on the ancilla qubit. That is, for a stabiliser operator, U_s,

$$U_\mathrm{s} E \left|\psi\right\rangle_\mathrm{L} = -E U_\mathrm{s} \left|\psi\right\rangle_\mathrm{L} = -E \left|\psi\right\rangle_\mathrm{L}.$$

The change in the eigenvalue from $+1$ to -1 can then be detected using the circuit shown in Fig. 2.5 with $U = U_s$.

It turns out that it is possible to discretise quantum errors such that correction of $E \in \{X, Z, XZ\}$ is sufficient to correct any error [31]. Therefore, we need only measure Z-stabilisers and X-stabilisers, where Z-stabilisers apply Z operations and X-stabilisers apply X operations on an even number of qubits. The Z-stabilisers detect X, or bit-flip, errors, which anticommute with the Z-stabiliser, and the X-stabilisers detect Z, or phase-flip, errors.

Returning to the repetition code example, we can use Z_0Z_1 and Z_1Z_2 stabiliser measurements, which are shown in Fig. 2.6(a) and Fig. 2.6(b), respectively, to find

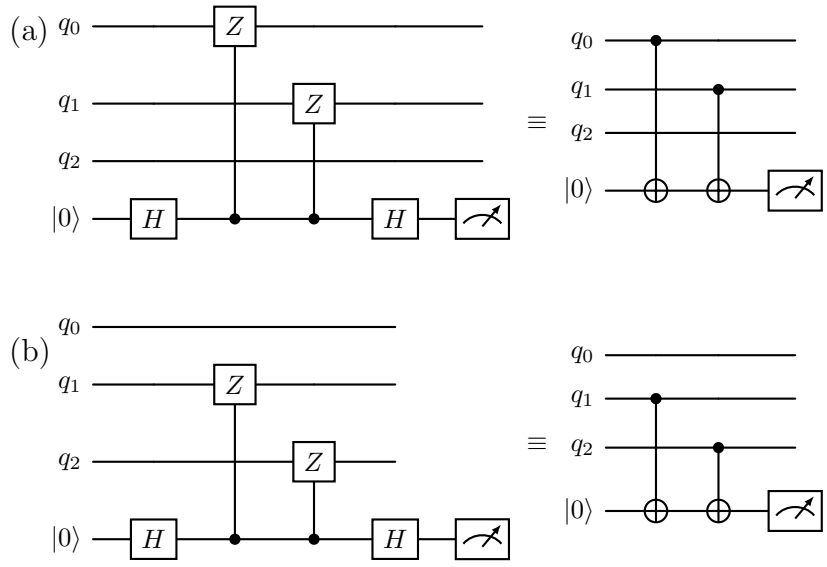


Figure 2.6: Z-stabiliser measurements (a) Z_0Z_1 and (b) Z_1Z_2.

a bit flip [6]. The measurement of $s_{z01} = Z_0Z_1$ yields the parity of the first two qubits while the measurement of $s_{z12} = Z_1Z_2$ yields the parity of the latter two. If we assume that at most one bit flip will occur then there are four possibilities:

1. $s_{z01} = s_{z_{12}} = 0$, indicating no error has occurred.

2. $s_{z01} = 1$, $s_{z_{12}} = 0$, indicating that a bit flip occurred on qubit q_0.

3. $s_{z01} = 0$, $s_{z_{12}} = 1$, indicating that a bit flip occurred on q_2.

4. $s_{z01} = s_{z_{12}} = 1$ indicating that a bit flip occurred on q_1.

We can represent the outcomes of both s_{z01} and $s_{z_{12}}$ collectively as a bitstring $s_z \in \{00, 01, 10, 11\}$, which we call the syndrome.

The repetition code with logical basis $\{|0\rangle_{\mathrm{L}} = |000\rangle\,, |1\rangle_{\mathrm{L}} = |111\rangle\}$ can only correct bit-flip errors, but if we instead use the basis $\{|0\rangle_{\mathrm{L}} = |+++\rangle\,, |1\rangle_{\mathrm{L}} = |---\rangle$, where $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$, we can construct and use the X-stabilisers X_0X_1, X_1X_2, in an identical fashion to the Z-stabilisers, and detect phase-flip errors. To correct either error or the product of the two, the bit-flip and phase-flip codes can be concatenated to give a code with the basis

states:

$$\begin{aligned} |0\rangle_{\mathrm{L}} &= |+\rangle_{\mathrm{L}} |+\rangle_{\mathrm{L}} |+\rangle_{\mathrm{L}} = \frac{(|000\rangle + |111\rangle)(|000\rangle + |111\rangle)(|000\rangle + |111\rangle)}{2\sqrt{2}} \\ |1\rangle_{\mathrm{L}} &= |-\rangle_{\mathrm{L}} |-\rangle_{\mathrm{L}} |-\rangle_{\mathrm{L}} = \frac{(|000\rangle - |111\rangle)(|000\rangle - |111\rangle)(|000\rangle - |111\rangle)}{2\sqrt{2}}, \end{aligned} \tag{2.12}$$

which is known as the Shor code. Many other codes are possible with varying properties, but each of them use the idea of taking stabiliser measurements on a subset of qubits and using them to build up a syndrome from which the position of errors can be inferred, allowing said errors to be corrected.

If we relax the need to correct errors and instead wish only to detect them then fewer resources are required and more errors can be handled. For example, if we wish only to detect errors with the three-qubit bit-flip repetition code then up to two errors can be detected while only one can be corrected. However, this comes at the cost of needing to restart the entire quantum circuit if any error is detected. Nonetheless, the reduced resource requirements in terms of qubits to handle a larger number of errors makes quantum error detection (QED) an attractive near-term prospect and we explore this idea further in Chapter 6.

Chapter 3

The dqc_simulator package

3.1 Introduction

Much of the work in this project relies on classical simulation but, prior to the commencement of this work, there was a dearth of software for easily simulating distributed quantum computers.

Several quantum network simulators exist [66–70] which have the capacity to simulate distributed quantum computers. Of these Netsquid [66] and SeQUeNCe [67] are the most commonly used in research because of the level of detail that they offer and their full-stack capabilities, with NetSquid seeing perhaps the most use. However, the general applicability of NetSquid and SeQUeNCe to any quantum network comes at the cost of making implementing specific networks, such as a distributed QC network, quite complicated.

For example, if one were to implement a complicated distributed quantum circuit with many inter-QPU, remote, gates in NetSquid, it would first be necessary to manually create connections between every single QPU in the network. Next, for each remote gate in the circuit, one would have to specify a small quantum circuit for both QPUs involved in the remote gate. Each sub-circuit would have to include all local operations up to that point and the various operations needed to implement the remote gate. The software must also be set up to send and wait for classical messages between both QPUs and communication qubits need to be referred to and managed manually. All of this must be done for every single remote gate in the circuit.

Far simpler distributed QC specific simulators also exist, namely CUNQA [71], Interlin-Q [72], DQCS [73], dqc-executor [74], and SimDisQ [75], although, CUNQA, DQCS and SimDisQ did not exist at the beginning of the work underlying this thesis. In any case, the existing distributed quantum computing simulators either do not allow the same level of physical detail to be simulated as NetSquid or are hard for users to access due to an absence of documentation or transparency. For example, dqc-executor, which is probably the closest to the simulator used in this work remains in the early stages of development and lacks documentation, making it hard to understand or utilise its capabilities. Moreover, it is intended to interface primarily with external files and lacks tools to help users work directly with pre-partitioned circuits. Similarly, we are unable to find a repository for SimDisQ or otherwise access its source code or documentation, and no mention is made in Ref. [75], where SimDisQ was introduced, of an easy way to implement time-dependent noise.

As such, there is a real need for a simulator which retains the flexibility and detail of NetSquid while automatically handling the more complicated details of distributed quantum computing implementation. dqc_simulator [76] is intended to do just that.

dqc_simulator is built on top of NetSquid and is intended to add functionality to the NetSquid package to make it easier to use for distributed quantum computing simulation. Advanced users, with knowledge of NetSquid, can use dqc_simulator as an extension to NetSquid but dqc_simulator is also intended to be largely self-contained and is suitable for users with no prior knowledge of NetSquid. The main features contributed by dqc_simulator are:

- An easy-to-use interface for specifying distributed quantum circuits.
- Automatic handling of communication qubits.
- An interpreter for commonly used communication subroutines such as remote gates.
- Extensible libraries of pre-made compilers and compilation tools.
- Automatic partitioning of monolithic quantum circuits between QPUs

- Parsing of .qasm files. It should be noted that this part of the package is implemented primarily using lightly modified nuqasm2 [77] source code. This usage is consistent with the open source license of nuqasm2.

The rest of this chapter is organised as follows. Section 3.2 provides a brief primer on object-oriented programming (OOP), to clarify the basic terminology relied on to properly discuss dqc_simulator. In Sec. 3.3, the relationship between dqc_simulator and NetSquid is explored in more detail and the design philosophy of dqc_simulator is discussed. Section 3.4 elucidates the functionality offered by dqc_simulator. Implementation details are provided in Sec. 3.5. A discussion of testing is given in Sec. 3.6. Finally, a summary of this chapter is provided in Sec. 3.7.

3.2 Object-oriented programming

dqc_simulator utilises a variety of object-oriented programming (OOP) principles in its implementation. Therefore, to understand dqc_simulator it is helpful to have a basic understanding of what OOP is.

As the name suggests, OOP involves centring code around abstract software entities called objects, which encapsulate data and functions, called methods. Objects may or may not correspond to an object in the real world. Objects are created from blueprints for a given type of object called classes. We say that an object is an instance of a class.

Classes can be related to each other in various ways. A class, C_1, can inherit the data and methods from another class, C_2, and extend the functionality of that class. We would then call C_2 a subclass of C_1. When C_2 inherits from C_1, we can say that C_2 *is* a C_1. For example, if we define a class called `Animal` and define a subclass `Horse`, then the sentence a `Horse` is an `Animal` is valid.

Another way of making classes is via composition. Composition relationships are defined by the idea that C_1 *has* a C_2. For example, we can define a class called `Car` which has an `Engine`. In such a relationship, the `Car` needs the `Engine` to be defined. It should also be noted that a class can be composed of several other classes.

A similar relationship, aggregation, is defined by the idea that C_1 *can have* a C_2. For example, a `Town` can have a `ShoppingCentre` but the `Town` and the `ShoppingCentre` also make sense separately from one another.

It is common to express these relationships graphically using a Unified Modelling Language (UML) diagram [1], as we do in Sec. 3.5.

3.3 dqc_simulator and NetSquid

The key motivation behind the creation of dqc_simulator is to make the underlying NetSquid [66] package much easier to use for distributed quantum computing applications while retaining its generality as much as possible. With this in mind, dqc_simulator was designed with what we call a 'have your cake and eat most of it containment' philosophy. This means that dqc_simulator is intended to be predominantly self-contained but is also intended to be compatible with code that predominantly uses native NetSquid. Moreover, when NetSquid already offers an easy way to do something then we do not offer an alterative. In this way, we attempt to minimise the amount of NetSquid knowledge required without restricting the ability of more advanced users to fully take advantage of NetSquid's versatility or introducing too much redundancy.

The maximal self-containment is achieved when dqc_simulator is used in the default way. This means that the user wishes to find out the output fidelity of a quantum algorithm conducted on a distributed quantum computer with simple, black box entangling connections between QPUs, in which entangled qubits with a certain, user-specifiable state, are sent to QPUs from some abstract central source. This is the default behaviour of dqc_simulator and no understanding of NetSquid is required to use it. A couple of NetSquid commands are required, which are relatively self-explanatory, but these can be copied verbatim from the code blocks introduced in Sec. 3.4 and no reference to the NetSquid documentation or deeper understanding of what the commands do is needed.

dqc_simulator also defines the protocols needed for emitting photons from QPUs and using a central BSM to distribute entanglement between those QPUs. However, to do so, a `MiddleHeraldedConnection` from the third-party package NetSquid-PhysLayer [78] is needed. It is hoped that, in the future, the open

source community will add to this default functionality, further reducing the need for users with other use cases to possess any knowledge of NetSquid.

When extending beyond the default behaviour, some knowledge of NetSquid is required. To gather simulation results other than the output fidelity, users can create their own `DataCollector` objects. Details on how to do this are available in the NetSquid documentation [79]. To use alternative inter-QPU connections, users can create their own using NetSquid's `Connection` class, as detailed in the NetSquid documentation [79], and should make their own subclasses of `PhysicalLayerProtocol`, defined in the `software` module of dqc_simulator. For the latter task, an understanding of NetSquid `Protocol`s would be useful. Further information about `PhysicalLayerProtocol` can be found in Sec. 3.5.

For more significant departures from the default behaviour, for example, if a user wishes to integrate distributed quantum circuits into other processes, then a more comprehensive understanding of NetSquid and its functionality is most likely required.

Nonetheless, even the default behaviour allows for a very broad range of applications, as detailed in Sec. 3.4.

3.4 Functionality

dqc_simulator is implemented in Python 3.9 and all of its functionality is accessible through the creation and running of Python scripts. The package consists of four subpackages, each with a conceptually different task. These are:

- **hardware**, which is used to emulate distributed quantum computing hardware. This subpackage defines general classes for the creation of a distributed quantum computer and a QPU compatible with distributed quantum computing. The hardware subpackage also defines more specific subclasses intended to emulate specific hardware systems and noise models useful for distributed quantum computing.

- **software**, which contains everything needed to define, compile and interpret quantum circuits on a distributed quantum computer.

- **qlib**, which is an extensible library of various predefined quantum objects, such as specific quantum states and quantum circuits. This makes working with commonly used quantum circuits, identities, gates and states more convenient.
- **util**, which contains helper functions useful for using or contributing to dqc_simulator.

The typical workflow for working with dqc_simulator is:

1. Specify the distributed quantum computer hardware to emulate using the hardware subpackage.
2. Specify the quantum circuit. This can either be a distributed quantum circuit or a monolithic one. In the latter case, some preprocessing, using the software subpackage, is required to distribute the circuit.
3. Input the quantum circuit to the interpreter defined in the software subpackage.
4. Initialise data collection using the util subpackage.
5. Start the interpreter.
6. Start the simulation using the `sim_run()` command from the underlying NetSquid package.

In the following subsections, we explore these steps in more detail.

3.4.1 Specifying hardware

When simulating distributed quantum computing, it would be possible to work at the circuit level and simply add in noise as additional fictitious gates. However, in doing so, we lose all of the details of the physical implementation. QPU qubit connectivity restrictions and locality must be manually enforced by the user and the modelling of networking processes like entanglement distribution becomes highly fictitious. Moreover, the impact of localised noise during physical processes cannot be understood and the latency contributions of many individual processes

must be amalgamated. It can also be harder to keep track of the positions of many different QPUs. As such, one cannot fully emulate and compare specific hardware in detail, which is a key use case for classical simulation.

NetSquid [66] natively offers extensive hardware emulation functionality. However, to retain generality, users are left to manually connect all QPU nodes within a given network and each node must be manually specified. dqc_simulator makes this much easier by allowing users to specify the quantum and classical topology that they want and have all nodes and connections be automatically generated.

In dqc_simulator, all of the emulated hardware needed to specify a distributed quantum computer is encapsulated by the `DQC` class, which is a subclass of NetSquid's `Network` class. There are a number of possible parameters that can be specified when instantiating a `DQC` object but the key options are what type of QPU to use, what type of quantum connection to use and the parameters that they should both have. With this information specified, a `DQC` object can then be easily created, for example as follows:

Code block 3.1: Defining a DQC object

```
import itertools as it

from dqc_simulator.hardware.connections import
↪ BlackBoxEntanglingQsourceConnection
from dqc_simulator.hardware.dqc_creation import DQC
from dqc_simulator.hardware.quantum_processors import
↪ NoisyQPU
from dqc_simulator.qlib.states import werner_state

# Defining QPU
qpu_type = NoisyQPU
kwargs4qpu = {'p_depolar_error_cnot' : 1e-03,
              'single_qubit_gate_error_prob' : 2e-05,
              'meas_error_prob' : 3e-03,
              'comm_qubit_depolar_rate' : 0.06,
```

```
                'proc_qubit_depolar_rate' : 0.05,
                'single_qubit_gate_time' : 135 * 10**3,
                'two_qubit_gate_time' : 600 * 10**3,
                'measurement_time' : 600 * 10**4,
                'num_positions' : 10,
                'num_comm_qubits' : 2}

# Defining connection
entangling_connection_type =
↪  BlackBoxEntanglingQsourceConnection
F_werner = 0.9
kwargs4conn = {'delay' : 1e9/182, #in ns. 182 Hz.
               'state4distribution' : werner_state(F_werner)}

# Defining the DQC
num_qpus = 3
quantum_topology = [(0, 1)]
classical_topology = list(it.combinations(range(3), 2))
dqc = DQC(entangling_connection_type, num_qpus,
          quantum_topology, classical_topology,
          qpu_class=qpu_type,
          **kwargs4qpu, **kwargs4conn)
```

which produces the `DQC` object shown in Fig. 3.1.

The code in code block 3.1 can be broken down into three simple steps.

1. We define the type of QPU that we wish to use and the parameter values it should have.
2. We do the same thing for the entangling connection.
3. We specify the parameters of the network itself and instantiate the `DQC` object.

For the first step, `dqc_simulator.hardware.quantum_processors` module provides the `QPU` class, which defines some common features that QPUs in a

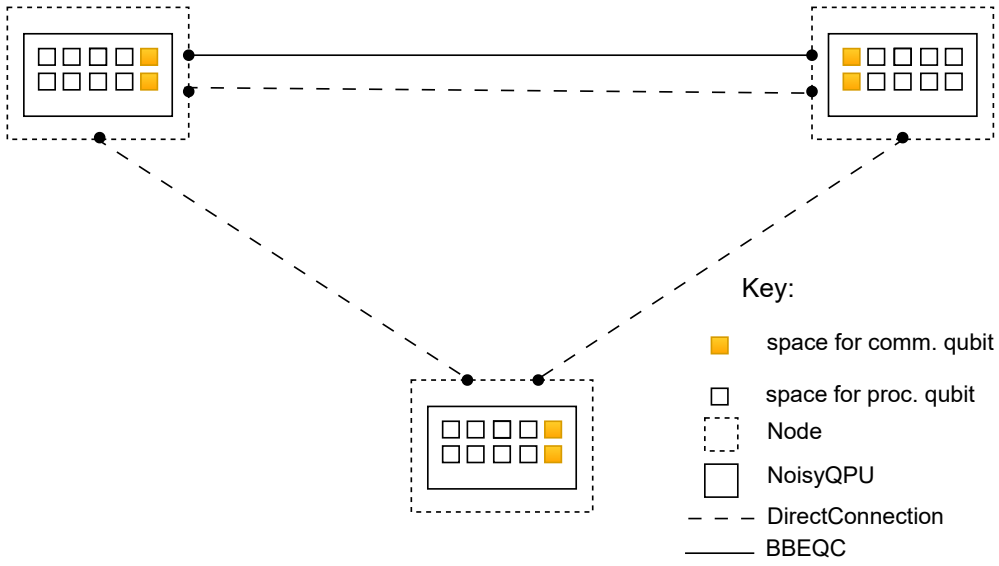


Figure 3.1: A simplified diagram of the `DQC` object created by code block 3.1. The acronym BBEQC in the key is short for `BlackBoxEntanglingQSourceConnection`. Comm. is short for communication and proc. for processing.

distributed quantum computer should have, such as ways to keep track of: which qubits are communication and processing qubits; whether the QPU has any communication qubits free for use; and whether the QPU in question is currently hosting half of an ebit.

Users wishing to emulate specific hardware can then create their own sublcasses of the QPU class while those whose focus lies elsewhere can use the pre-made `NoisyQPU` subclass which accommodates the Clifford + T native gate set and the capacity to specify single or two-qubit unitaries as NumPy [80] arrays. `NoisyQPU` has all-to-all connectivity between qubits and applies the noise models described in Sec. 6.4.2 of Ch. 6 subject to user-specifiable parameters. The timings of gates and measurements are also user-specifiable. As dqc_simulator is open source, it is hoped that additional subclasses will be added to the `quantum_processors` module by users in the future. It should be noted that it is not necessary for the user to actually instantiate the `QPU`, class, or any subclasses thereof, themselves as this is done behind the scenes by the `DQC` class. Instead, the user just specifies what class to use and the parameters separately.

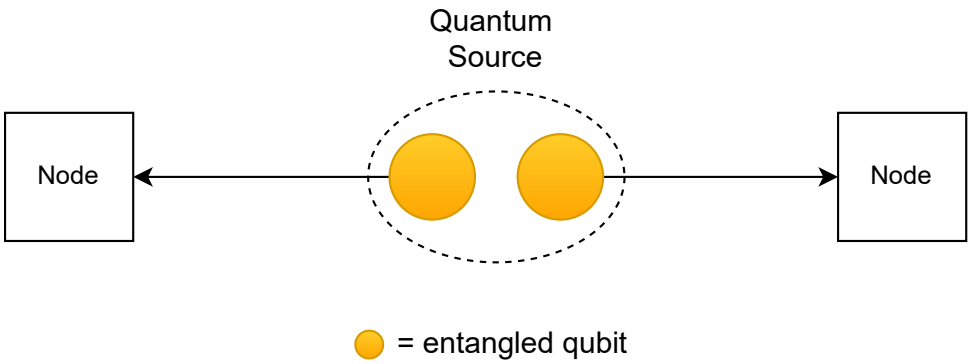


Figure 3.2: A simplified conceptual diagram of the default quantum connections provided by dqc_simulator.

It is also possible for advanced users familiar with NetSquid to specify a list of more complicated network nodes that contain more than just a QPU in the same spatial network location. This can be done by specifying a list of NetSquid `Node` objects. Details around how `Node` objects work and the other type of hardware require more interaction with NetSquid itself [66] and exceed the scope of this work. By default, `Node` objects are created in the background when a `DQC` object is created but a dqc_simulator user need only refer to their names, which have the form `'node_i'` where i is a natural number index ≥ 0, and the `qmemory` attribute to access the QPU. For example, the command `dqc.get_node('node_0').qmemory` will recover the first QPU from in the `dqc` network instantiated using code block 3.1.

Step two is very similar. The user can either create their own subclass of NetSquid's `Connection` class or use one of the subclasses already provided by the `dqc_simulator.hardware.connections` module. The current options are both of the form depicted in Fig. 3.2, which shows a simplified conceptual diagram of the way that the default connections both work. As illustrated in Fig. 3.2, each of the default connections contains a central source of entangled qubits in a user-specified state and distributes the entangled qubits directly to the network nodes, abstracting away details of exactly how entanglement is generated and the transduction process between flying and communication qubits. These connections

can be thought of as black boxes generating qubits in a certain state, and noise can be modelled by defining that state to be noisy.

The first of the default connections, `ProbabilisticQSourceConnection`, retains NetSquid's formalism agnosticism and so requires the user to specify several two-qubit states that the ebits could be in and the probabilities of the ebit being in each of those states. The second of the default connections, `BlackBoxEntangling QsourceConnection`, is similar but exchanges formalism agnosticism for a simple interface in which the user need only specify a single density matrix for the ebit. This allows users to make use of the pre-specified states defined in the `dqc_simulator.qlib.states` module for such a purpose. For example, the predefined `werner_state` density matrix used in the example above. Many alternative pre-defined `Connection` subclasses can already be found in various third-party open-source NetSquid snippets, such as the NetSquid-PhysLayer snippet [78], which offers a more physically detailed connection that generates entanglement between flying qubits generated by QPUs via a central BSM. However, it should be noted that a subclass of `PhysicalLayerProtocol` from the software subpackage of dqc_simulator must be specified if the connection functions fundamentally differently from the ones native to dqc_simulator. More details on this are provided in Sec. 3.5. Subject to this constraint dqc_simulator is compatible with arbitrary NetSquid `Connection` objects. Classical connections can also be specified in a similar fashion if desired. By default, an optical fibre is used with length equal to the `node_distance` parameter of the `DQC` class. The default `node_distance` is 2m.

Finally, in step three, the higher level networking details are established, such as the number of QPUs, and the topology of the network with respect to the classical and quantum connections between QPUs and this, along with all the information from the previous steps, is inputted to create a `DQC` object.

3.4.2 Specifying quantum algorithms

The cost of a more detailed full-stack simulation, where hardware is explicitly included, is that the specification of quantum algorithms becomes complicated. Rather than one overall quantum circuit for the entire algorithm, we now have

many smaller quantum circuits which need to interact. As it does not focus on distributed quantum computing applications, native NetSquid has no automatic way for dealing with this and so specifying quantum circuits is complicated and the user must start from scratch each time a new algorithm is considered. A significant part of the motivation for developing dqc_simulator was solving this problem.

Many users, for example those focussed on hardware or networking aspects of distributed quantum computing research, may wish to think only in terms of monolithic quantum circuits of a form that could be run on a monolithic, single-QPU device. This is conceptually simple and allows users to utilise a whole host of external resources such as libraries of pre-defined quantum circuits like MQT Bench [4], popular software development kits (SDKs) like Qiskit [81] or quantum assembly languages like openQASM 2 [82]. Many more of these exist for monolithic QC than distributed QC and many QC researchers may already be familiar with them. On the other hand, some users may be specifically investigating partitioning between QPUs and/or other facets of distributed quantum computer compilation and so want to have a way of writing a single quantum circuit per algorithm while retaining the capacity to explicitly specify the partitioning of qubits between QPUs and the sequencing of operations. dqc_simulator aims to fulfil both needs.

Starting from a pre-partitioned circuit

Pre-partitioned circuits can be easily specified as lists of gate tuples, which are easy to create and manipulate in Python for users looking to perform their own partitioning. Gate tuples can take on one of three forms that correspond to the type of gate being specified. These are:

1. `(gate_instr, qubit_index, node_name)` for single-qubit gates
2. `(gate_instr, qubit_index0, node_name0, qubit_index1, node_name1)` for local two-qubit gates.
3. `(gate_instr` or `gate_instrs, qubit_index0, node_name0, qubit_index1, node_name1, scheme)` for remote two-qubit gates.

where

- `gate_instr` is an object of the NetSquid `Instruction` class or, for advanced users with more understanding of NetSquid, a tuple containing an `Instruction` object and a NetSquid operator object can be used to specify arbitrary unitary gates. Instructions have standard names of the form `INSTR_<symbol>` where `<symbol>` is a circuit notation symbol for the gate, such as H for hadamard. The main exceptions to this are `INSTR_INIT`, which initialises qubits, and `INSTR_MEASURE`, which performs a measurement (in the computational basis by default). The Hermitian conjugates of many common gates are included in the `dqc_simulator.qlib.gates` module and can be referred to by appending the suffix `_DAGGER`. For instance, the $T^\dagger$ gate is called `INSTR_T_DAGGER` and must be specially imported from the `dqc_simulator.qlib.gates` module.

- `qubit_index` or `qubit_index_i` for $i \in \{0, 1\}$ is either a natural number index indicating the position of the qubit or -1 to refer to a communication qubit (which specific communication qubit will be used is decided automatically behind the scenes). For convenience, the indices of the processing qubits for a `QPU` object can be accessed by the `processing_qubit_positions` of a QPU. Starting with a `DQC` object, `dqc`, one can use the command `dqc.get_node(node_name).qmemory.processing_qubit_positions` where `node_name` is defined in the next bullet.

- `node_name` or `node_name_i` is the name of the network node containing the qubit referenced with the preceding `qubit_index`.

- scheme is a string containing the type of remote gate to use. The options are 'cat', '1tp', '2tp' and 'tp_safe', which refer to cat-comm, 1TP, 2TP and TP-safe, respectively.

- `gate_instrs` is a list of gate tuples, specified in the manner described above, defining local gates that should be applied collectively between the entangling and disentangling operations of cat-comm or the first and second teleportation of 2TP or TP-safe. This allows compound remote gates to be defined, as is done in some compilation schemes such as the one introduced in Ref. [21].

Much of dqc_simulator is also compatible with further extending this functionality to larger multi-qubit gates by repeating the `qubit_index`, `node_index` pattern for all qubits involved in the gate but this functionality has not yet been fully tested.

This gate tuple formalism allows an entire distributed quantum circuit to be specified as a single list while clearly defining all partitions in the circuit and how remote gates will be broken down into primitives. For users who wish to create their own partitions or simply want an easy way to directly define distributed quantum circuits at a higher level then this formalism should offer all of the functionality they need.

With a distributed quantum circuit defined in this way, it is then trivial to run the quantum circuit on the hardware previously defined in code block 3.1. For example, the code

Code block 3.2: Running a partitioned circuit

```
import netsquid as ns
from netsquid.components import instructions as instr

from dqc_simulator.software.dqc_control import
↪  DQCMasterProtocol

# Defining the gates
gate_tuples = [(instr.INSTR_INIT, range(2, 5), 'node_0'),
               (instr.INSTR_INIT, range(2, 5), 'node_1'),
               (instr.INSTR_INIT, range(2, 5), 'node_2'),
               (instr.INSTR_H, 2, 'node_0'),
               (instr.INSTR_CNOT, 2, 'node_0', 2, 'node_1',
               ↪  'cat')]

# Running the circuit
interpreter = DQCMasterProtocol(gate_tuples, nodes=dqc.nodes)
interpreter.start()
ns.sim_run()
```

runs a short quantum circuit on the `dqc` object created with code block 3.1. By default, a simple greedy compiler is used, defined by the function `sort_greedily_by_node_and_time`, for when there are two QPUs, and the `sort_many_qpus_greedily_by_node_and_time` function, when more than two QPUs are present. Both functions can be found in the `dqc_simulator.software.compilers` module.

Defining your own compilation

For some users, such as those who wish to research the scheduling aspects of compilation, it is essential to be able to define your own compiler. Users wishing to do so can define a compilation function of their own and input the function to the interpreter using the `compiler` option of `DQCMasterProtocol`. Although, as is typically the case for compilation, some more understanding of the implementation than was previously required may be useful, all that is essential to understand for such a user is the forms that the input and output of such a compilation must take. Therefore, discussion of the implementation details is deferred to Sec. 3.5.

To be compatible with `'dqc_simulator'` a compiler function must convert from the list of tuples defined in Sec. 3.4.2 to the lower level representation that the interpreter uses behind the scenes. In this lower level representation, the circuit is broken down by node and in addition to gates, we also think in terms of primitives, which are subroutines that can include gates, classical logic, and classical communication. The different types of primitives, defined at the time of writing, and what they do are summarised in Tab. 3.1.

The lower level representation is a dictionary whose keys are node names and whose values are lists of lists of gate tuples. This can be thought of as a lookup table with the structure shown in Tab. 3.2. Each sublist within the overall list indicates a time slice and the gate tuples defined within it will be applied greedily. In this lower level representation of what is going on, the gate tuples have a different form from the one defined in Sec. 3.4.2, to reflect the fact that they now describe what goes on in a specific node rather than the behaviour of a gate as a whole.

Label	Type	Purpose
'logged'	Single-qubit	Saves result of operation or subroutine (eg, measurement result)
'distribute_ebit'	Miscellaneous	Handles ebit distribution for one node.

(a) Single-qubit and miscellaneous primitives

Label		
Scheme	Role	Purpose
'cat'	'entangle'	QPU A's share of cat-entanglement in Fig. 4.1(a).
'cat'	'correct'	QPU B's share of cat-entanglement in Fig. 4.1(a).
'cat'	'distentangle_start'	QPU B's share of cat-disentanglement in Fig. 4.1(a).
'cat'	'disentangle_end'	QPU A's share of cat-disentanglement in Fig. 4.1(a).
'tp'	'bsm'	QPU A's part of teleportation in Fig. 4.1(b).
'tp'	'correct'	QPU B's part of teleportation in Fig. 4.1(b). Reserves comm-qubit.
'tp'	'correct_manually'	'correct' with manual specification of comm-qubit enabled.
'tp'	'correct4tele_only'	'correct' but does not reserve comm-qubit.
'tp'	'swap_then_correct'	QPU A's part of the second teleportation in Fig. 4.1(d)

(b) Remote gate primitives

Table 3.1: The primitives understood by the 'dqc_simulator.software.dqc_control.DQCMasterProtocol' interpreter.

Key	TS0	TS1	TS2	...
`'node_0'`	[...]	[...]	[...]	...
`'node_1'`	[...]	[...]		
`'node_2'`	[...]	[...]	[...]	...
⋮	⋮	⋮	⋮	...

Table 3.2: Visualisation of the lower level representation that should be produced by a compiler. TS stands for time slice. [...] represents a list of gate tuples. The curtailing of the `'node_1'` row symbolises the fact that a different number of time slices can be used for each node provided that remote gate primitives corresponding to the same remote gate occur on the same time slice.

The key idea is that the primitives for a given multi-node operation, such as a remote gate, must occur at the same time slice to allow them to be correctly interpreted. Within a given remote gate a given node should at most do the primitives needed for a single remote gate and any preceding local gates, to prevent scheduling issues. Unfortunately, this means that remote gates involving a specific node must be performed sequentially, which does potentially restrict users from finding a completely optimal compilation solution for nodes with large numbers of communication qubits which could do more remote gates in parallel. It is hoped to remove this limitation in a future version of dqc_simulator. However, large numbers of communication qubits are unlikely in near term devices and aside from this constraint, users may impose arbitrary scheduling by splitting the circuit into more or fewer time slices as they deem fit.

For the lower level representation shown in Tab. 3.2, the tuples have the form:

1. `(gate_instr, qubit_index)` for single-qubit gates
2. `(gate_instr, qubit_index, primitive_type)` for single-qubit primitives
3. `(gate_instr, qubit_index0, qubit_index1)` for local two-qubit gates
4. Either
   (a) `(qubit_index, other_node_name, scheme, role)` OR
   (b) `(other_name_name, scheme, role)`

for remote gate primitives

where `primitive_type` is a string label indicating the type of primitive; `other_node_name` is the name of the other node involved in the remote gate; scheme is the type of remote gate, which can be `'cat'` or `'tp'` for cat-comm and any type of TP-comm respectively; and `'role'` is a label for the type of primitive, with allowed values shown in Tab. 3.1. All other terms have the previously defined meanings.

Starting from an openQASM 2.0 file

On the other end of the spectrum from users looking to work on compilation are those who wish to understand the implications of hardware choices or perform noise analysis. Such users typically don't care at all about the details of compilation and may be simply looking to test a given hardware setup or noise model on a wide variety of quantum circuits. As a variety of benchmarking suites, such as MQT Bench [4], are written in openQASM 2.0 and the openQASM 2.0 format is well known to many quantum computing researchers, it is helpful to be able to work with .qasm files.

When working with .qasm files, the aim is to go from the monolithic quantum circuit represented in the .qasm file to a list of gate tuples of the form defined in Sec. 3.4.2, which represent a partitioned quantum computer. This is easily done using the code:

Code block 3.3: Converting a .qasm file to gate tuples

```
from dqc_simulator.software.qasm2gate_tuples import
↪  qasm2gate_tuples

# assuming that working directory is repo root
filepath = ('./src/dqc_simulator/software/tests/unit_tests/'
            'ghz_indep_qiskit_5.qasm')
include_path =
↪  './src/dqc_simulator/software/tests/unit_tests/'
scheme = 'cat'
```

```
gate_tuples = qasm2gate_tuples(dqc, filepath, scheme,
                               include_path=include_path)
```

where I have assumed that the `dqc` object has already been initialised using code block 3.1 and that the working directory is the root of the `'dqc_simulator'` repository. If the latter criterion is not met, then the `filepath` and `include_path` must be adjusted accordingly. It should also be noted that measurements are currently ignored because in monolithic circuits these typically occur at the end of the circuit and NetSquid and dqc_simulator have alternative ways of interpreting simulation results. This may change in future versions.

Having run code block 3.3 one can then proceed exactly as discussed in Sec. 3.4.2.

3.4.3 Recording results

NetSquid defines very simple and convenient ways to inspect the state of the quantum system being simulated. This is done using the `DataCollector` class, which can inspect the state of the system in the formalism of choice during simulations. To this, dqc_simulator adds a useful helper function to create a specific `DataCollector` that can be used for experiments in which the figure of merit is the fidelity at the end of the circuit relative to a desired output state, expressed as a ket vector or density matrix using a NumPy [80] array. For more arbitrary experiments, the user must define their own `DataCollector`. We refer such users to the NetSquid documentation [79].

Using the interpreter defined in code block 3.2, data collection is set up as follows:

Code block 3.4: Preparing a data collector for use.

```
import numpy as np

from dqc_simulator.util.helper import get_data_collector
```

```
# Retrieving QPU nodes from DQC
node_0 = dqc.get_node('node_0')
node_1 = dqc.get_node('node_1')

qubit_indices_2b_checked = [(2, node_0), (2, node_1)]
desired_state = np.sqrt(1/2) * np.array([[1],[0], [0], [1]])
dc = get_data_collector(interpreter,
↪  qubit_indices_2b_checked,
                        desired_state)
```

The data collector, `dc`, should be created before the simulation is run and will monitor results of the simulation.

After the simulation is finished, the collected results can then be accessed in the form of a pandas dataframe [83] using the `dataframe` attribute of the `DataCollector` class via the command `dqc.dataframe`.

If desired, this is easy to convert to an Excel or CSV file as follows:

Code block 3.5: Converting pandas dataframe to Excel or csv file

```
# For exporting to Excel
filename = '<path>/results.xlsx' # replace <path> with
↪  desired path to file
results.to_excel(filename)

# For exporting to csv
filename = '<path>/results.csv'
results.to_csv(filename)
```

3.5 Implementation details

dqc_simulator is written using Python 3.9 and extensively uses the NetSquid [66] Python package, which in turn uses the PyDynAA [66, 84] discrete event simulator. The key features of the software architecture are graphically depicted

in Fig. 3.3, which also illustrates the key relationships between dqc_simulator, NetSquid and PyDynAA.

The implementation decisions most specific to dqc_simulator, as opposed to the underlying NetSquid package [66], occur in the `dqc_control` and `physical_layer` modules in the `software` subpackage. These two modules work together to run distributed quantum circuits on emulated hardware.

When describing quantum algorithms it is convenient to think in terms of one object, a quantum circuit, that describes the entire algorithm. However, from a networking perspective we want to enforce some sense of locality, as we are considering multiple spatially distinct locations, or nodes, which take some time to communicate with one another. To reconcile these two opposing notions, we use an omnipotent abstract protocol, `DQCMasterProtocol`, to enforce some basic network-wide scheduling and inter-node synchronisation but implement almost all of the functionality separately on each node, using explicit communication between nodes when necessary.

As discussed in Sec. 3.4.2, scheduling is imposed by splitting the overall distributed quantum circuit into time slices. Each node knows what it should do at every time slice but it does not know when to start a given time slice. Therefore, at the beginning of each time slice, `DQCMasterProtocol` sends each node a signal to let it know that it can begin carrying out the instructions it has for the next time slice. This is the only type of signal that `DQCMasterProtocol` sends and each node only sends one type of signal back to the `DQCMasterProtocol`, to let it know it has finished carrying out the operations for that time slice. Therefore, crucially, the nodes do not use `DQCMasterProtocol` as an intermediary for instant messaging. The signalling to and from `DQCMasterProtocol` is shown in Fig. 3.4

While it may seem natural to think of what is happening as a central classical controlling node messaging several QPU nodes in a network, the instantaneous nature of the signalling means that this is not really what is being modelled. Instead, it should be thought of as a simple abstraction of scenarios in which every node in the network is given everything it needs to do on all time slices in advance and then the performance of each time slice is synchronised somehow. For example, it could be that a time slice represents a specific length of time established during the initial preprocessing step and that the clocks of all nodes are synchronised in

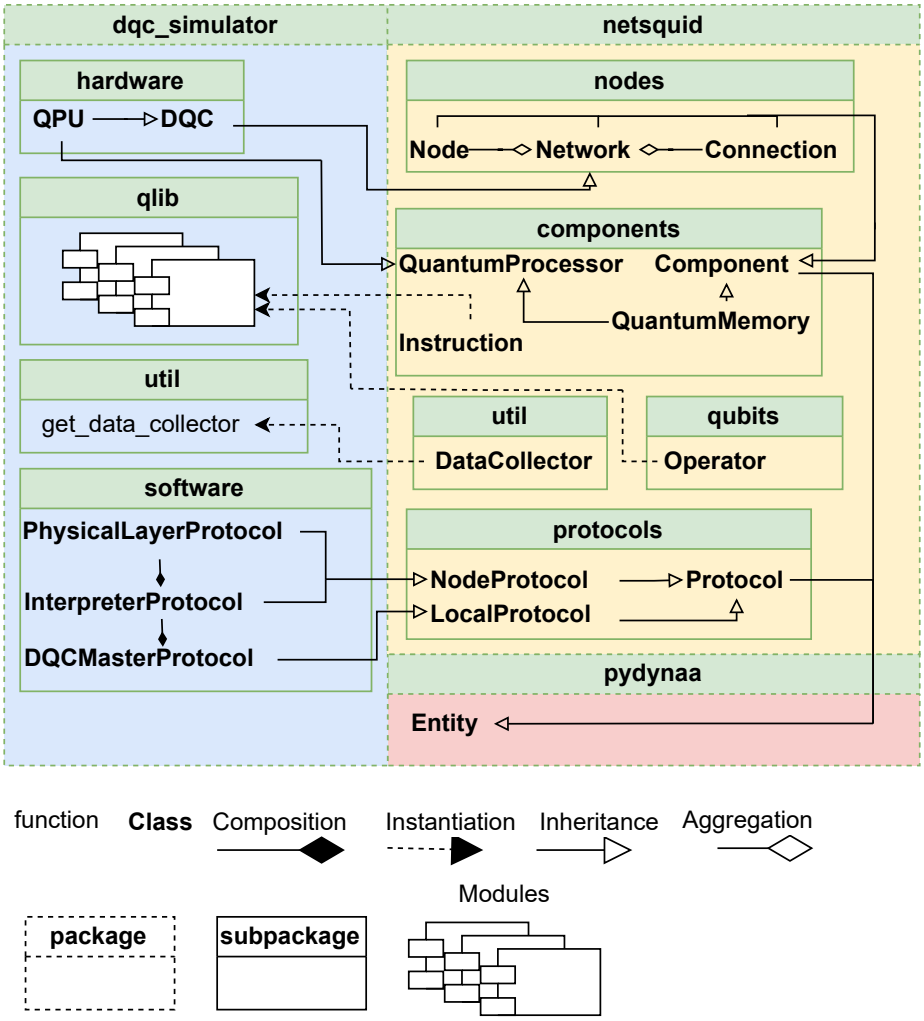


Figure 3.3: A simplified software architecture of dqc_simulator. Only the most important classes and functions are shown. We use the arrows conventional for UML diagrams [1]. Let class 1, C_1, be the class, at the tail of the arrow and class 2, C_2 be the class at the head of the arrow. Composition indicates that C_2 **has** a C_1. Inheritance means that C_1 **is** a C_2 and extends its functionality. Aggregation means that C_2 **can have** a C_1 but C_1 and C_2 can exist independently of each other. Instantiation means that the object at the head of the arrow creates an instance of the class C_1.

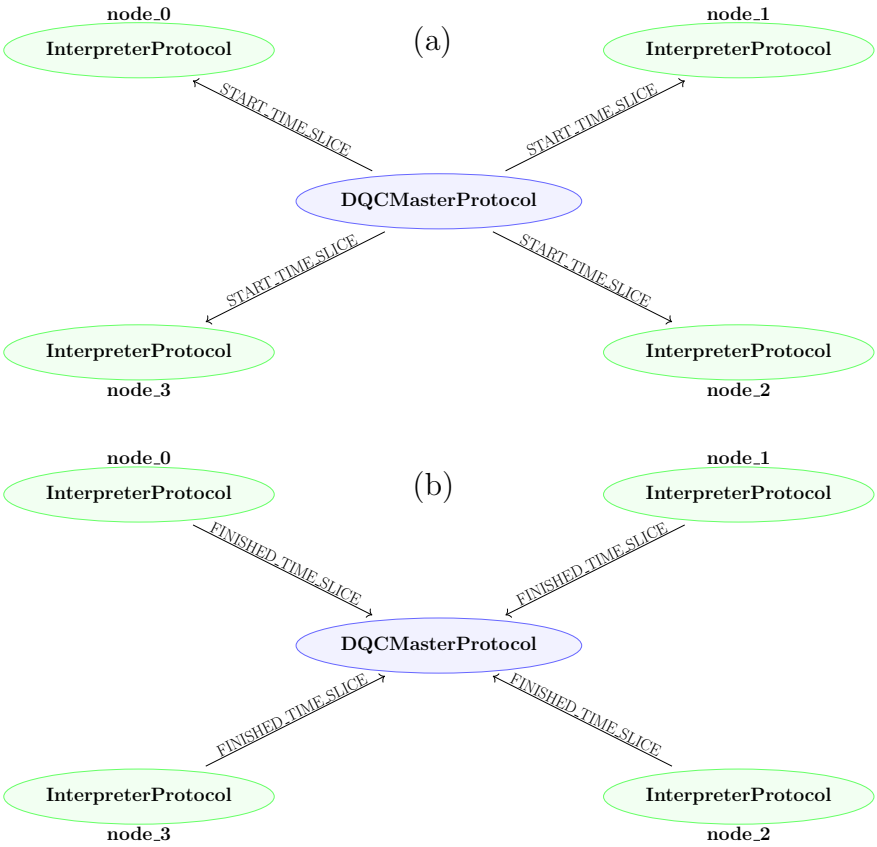


Figure 3.4: Signalling (a) from and (b) to `DQCMasterProtocol`.

advance. `DQCMasterProtocol` abstracts away the complicated details of such a process making development much easier. Actual physical networking tasks, like messaging between QPUs are handled separately and locality is strictly enforced for such tasks.

For handling the physical networking tasks, each network node has its own interpreter called `InterpreterProtocol` which carries out the gates, measurements and primitives for that node. `InterpreterProtocol` is where each primitive is defined and run in the simulator. It is the interpreter, that runs the instructions from the relevant node entry in the lookup table.

When `InterpreterProtocol` needs to do a primitive that involves communication with another node then it must interact with the communications protocol stack [85]. Protocol stacks are a way of abstracting the exact protocols that a node should do when communicating with another node, splitting the overall task of communication into several abstract sub-tasks or layers [85]. Each layer uses functionalities provided by the layer immediately beneath it and provides functionality to the layer above it. In the quantum context, layers layers also sometimes provide functionality to the layer directly below them. dqc_simulator takes conceptual inspiration from the Wehner protocol stack [86, 87], which is shown in Fig. 3.5(a). The Wehner stack is the protocol stack most naturally compatible with NetSquid, as it was invented by the same research group that developed NetSquid. It consists of five layers: the physical layer, which is hardware specific and attempts to generate entanglement between two nodes in discrete time slots; the link layer which decides whether the physical layer should attempt entanglement generation and manages the heralded results of entanglement generation attempts; the network layer, which manages long-distance entanglement generation, taking care of things like entanglement swapping; the transport layer, which moves quantum information from processing qubits around using processes like teleportation; and the application layer, which requests the movement off quantum information [85]. However, we explicitly consider only the first two layers, the physical and link layers, of the five suggested in Refs. [86, 87] and swap their ordering as shown in Fig. 3.5(b). Moreover, our implementation of each layer differs quite significantly from the exact protocols proposed in Refs. [86, 87], which focus on scenarios where entanglement is created using BSMs between

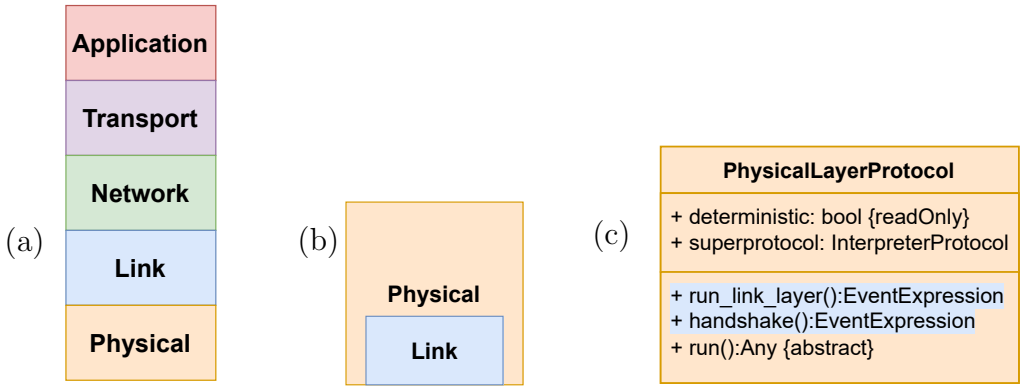


Figure 3.5: (a) The Wehner protocol stack. (b) The explicit protocol stack used in dqc_simulator. (c) Class diagram showing the implementation of the physical and link layers. The class name appears in the topmost box, the attributes in the middle box and the methods in the bottom box. Attributes are quoted with the form 'name: type' and methods with the form 'name:return type'. Many attributes are omitted for overall clarity. Curly braces, , indicate a property or constraint and the abstract label indicates that the method must be overwritten by all subclasses. Bool is short for boolean and EventExpression is from the PyDynAA.core module. Functionality from higher layers from the Wehner protocol stack is implicitly included in dqc_simulator but is not explicitly segregated into the same layered structure as used in the Wehner stack.

adjacent nodes. Another key difference between our implementation and that of Refs. [86, 87] is that we encapsulate the link layer within the physical layer. We can say that the physical layer has a link layer, which is not true of Refs. [86, 87].

Both the physical and link layers are concerned with generating entanglement between QPUs on different nodes and refer to protocols that should be done on each node involved in the generation of an entangled link. We broadly define the link layer as a protocol or protocols done on one node that contribute to entanglement generation and that we do not expect to change when the type of hardware used for the QPU or the connection between nodes changes. The physical layer is then the hardware specific parts of entanglement generation. We encode the baseline functionality of the link layer into methods defined within the class `PhysicalLayerProtocol`, as shown in the simplified class diagram of the PhysicalLayerProtocol shown in Fig. 3.5(c). The physical layer for a given hardware setup is then implemented as a subclass of `PhysicalLayerProtocol`. The subclass calls the aforementioned hardware-independent baseline methods inherited from `PhysicalLayerProtocol` and builds upon the baseline by adding

hardware-dependent functionality with more hardware specific actions that should be taken by each node to host entanglement.

In our implementation, two protocols, implemented as methods of `PhysicalLayerProtocol`, are used to define the link layer. These are `run_link_layer` and `handshake`. The two protocols are kept separate because we are unsure if all users will always wish to do the handshake protocol. However, in most cases both protocols will be run sequentially prior to implementing the physical layer protocol.

`run_link_layer` waits for an entanglement request from the `InterpreterProtocol` on the node it is running on. This request contains the information needed to do the entanglement such as what communication qubits to use and the number and type of ebits to generate. It also indicates whether the node it is running on is the 'client' or 'server' for the entanglement distribution. `handshake` then checks both nodes involved in an entangled link are ready for entanglement to be distributed between them. The handshake protocol starts with the client sending a "READY4ENT" message to the server. The client waits for a set duration, T_t, on a response from the server. If it receives a response of "READY4ENT" within T_t then the handshake is considered successful and the physical layer protocol is allowed to proceed. Otherwise, the client resends the "READY4ENT" message. This process repeats until the handshake protocol is successful. The possible ways the protocol can proceed are summarised in Fig. 3.6.

Right now, two types of physical layer are defined, which are alluded to in Sec. 3.3. The first, `AbstractCentralSourceEntangleProtocol`, is for working with abstract, black-box, central-source connections such as `ProbabilisticQSourceConnection` and `BlackBoxEntanglingQsource`. After the link layer protocols have completed, `AbstractCentralSourceEntangleProtocol` sends an entanglement request directly to the central black-box source of qubits and waits to receive those qubits. It assumes that the node it acts on receives an already entangled qubit from the connection and simply puts that qubit on the first unoccupied communication qubit position in the memory. A diagram of the physical situation that `AbstractCentralSourceEntangleProtocol` models is shown in Fig. 3.7(a)

The other physical layer option, `MidpointHeraldingProtocol`, causes the emission of a photon from a communication qubit, as shown in Fig. 3.7(b). This

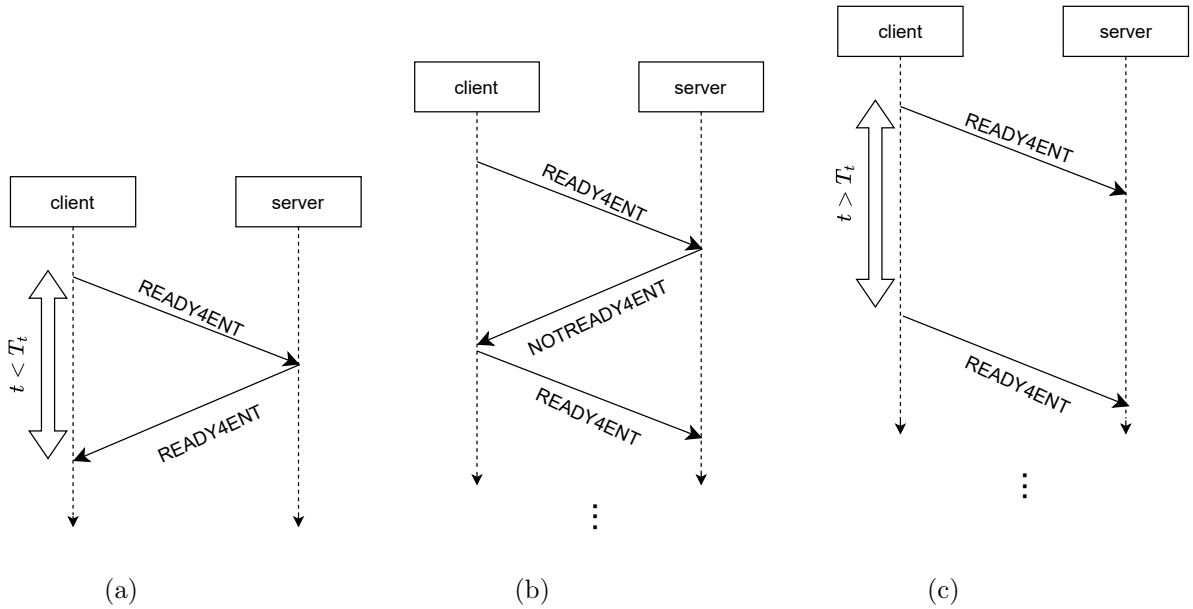


Figure 3.6: The handshake protocol if (a) the server is ready when it receives the message from the client, (b) the server is not ready when it receives the message from the client, (c) timeout occurs before the client receives any message. Tim increases from top to bottom.

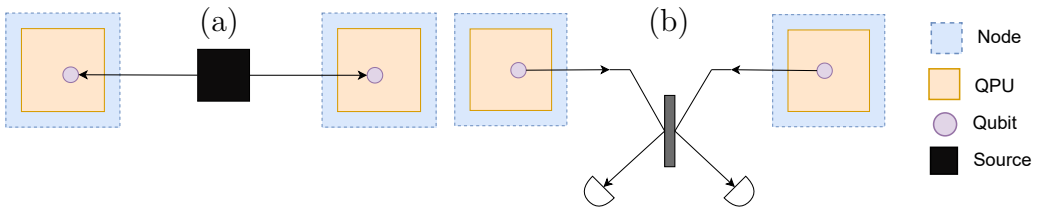


Figure 3.7: The specific physical layer protocols included with dqc_simulator: (a) `AbstractCe` ⌋ `ntralSourceEntangleProtocol` and (b) `MidpointHeraldingProtocol`.

photon is modelled as a qubit on a fictitious memory position in the QPU which is set aside for photons. The memory position is fictitious only in the sense that it is not intended to represent a physical static location but is convenient for modelling. The photon is sent to the connection, in which a BSM should be done, and `MidpointHeraldingProtocol` waits on the BSM results. If a coincident click is obtained in the BSM, indicating both measurement results constituting the BSM were obtained simultaneously, then the BSM is considered a success and the protocol is allowed to proceed. In that case, local gates may be applied depending on the measurement result in exactly the same way as for teleportation, see Fig. 4.1(b). If the BSM instead fails then another photon is emitted and the whole protocol is restarted. This entire process is repeated until a successful BSM occurs or the maximum number of iterations is exceeded.

Collectively, `DQCMasterProtocol`, `InterpreterProtocol` and the link and physical layers enable the running of distributed quantum circuits on emulated distributed QC hardware. They are the engine room for dqc_simulator and provide the primary constraints on how it works.

3.6 Testing

Simulators are an especially challenging type of software to test. It is necessary to ensure that not only can a simulation be implemented but that it accurately models the physical reality. To meet these criteria, different forms of testing are carried out.

Firstly, unit testing is done in cases where the behaviour of a function or method should give a deterministic output. This is the case for example with the compilation and parsing functionality of dqc_simulator where for a given input, the correct output can be ascertained.

It is also tested that the output of the simulation as a whole gives the expected result for specific simple, physical scenarios in which analytical results can be obtained. For example, it is tested that a remote CNOT gate implemented using various schemes gives the correct output for specific input states.

Finally, physical sense-making tests are done, in which scenarios are considered for which the exact result may be unknown but some constraints can be put on it

and tested for. For example, the output fidelity of a large circuit should be less than one in the presence of noise but must always remain greater than zero.

Much of the final kind of testing was linked to specific experiments conducted during the course of this research project and so are segregated from the dqc_simulator package itself. It remains a matter of future work to decide on at least some example use cases and reintegrate these into the simulation tests available to all users. Moreover, an easy way of running all tests at once remains to be implemented as right now they are split by module.

3.7 Conclusion

In this chapter, we have discussed the dqc_simulator tool which was used to obtain many of the results in this thesis. The development of dqc_simulator represents a significant proportion of the research time for this project and is one of the contributions to the field made in this project.

We have explored the functionalities and use cases for dqc_simulator as well as some of its limitations. In brief, dqc_simulator is most naturally used by those who wish to treat a distributed quantum circuit as a single object, much like they would with a monolithic quantum circuit, but still wish the details of the underlying quantum network to be explicitly modelled. This is especially useful for benchmarking tasks, such as the benchmarking of hardware or compilers but the ready integration of dqc_simulator with the underlying NetSquid libray [66] makes it an extremely flexible tool for asking a variety of questions about distributed quantum computers and quantum networks more generally. Further details have also been provided on the implementation of dqc_simulator.

It is hoped that dqc_simulator will see adoption by other researchers in distributed quantum computing research. To facilitate this, we have made dqc_simulator open source. The source code, documentation and installation instructions for dqc_simulator are available at Ref. [76].

Chapter 4

Noise analysis of single remote gates

4.1 Introduction

In this chapter, we consider what happens when a two-qubit gate must be carried out on qubits that exist on different QPUs. We call this a remote gate. Remote gates represent the key difference between monolithic and distributed quantum circuits and so understanding their behaviour is key to understanding the behaviour of QDCs as a whole. We make the following contributions to the understanding of remote gates:

1. We demonstrate the limitations of first-order error analysis relative to full classical simulation in the QDC context.

2. We develop analytical expressions for different types of remote gate subject to the assumptions that the input state is separable and that the only noise on the system is in the inter-QPU entanglement used to facilitate the remote gates.

3. We determine the relative impact of noise in inter-QPU entanglement, local gates, and decohering qubits (as time evolves) on the performance of individual remote gates.

4. We compare different approaches to inter-QPU (remote) gate operation by calculating the output fidelity for each approach.

The rest of the chapter is organised as follows. Firstly, we discuss the relevant literature in Sec. 4.2. Following that, the problem at hand is elucidated in Sec. 4.3. Then, the methods we use to analyse remote gates are introduced in Sec. 4.4. The results of that analysis are presented in Sec. 4.5. Finally, we summarise our findings and discuss their implications in Sec. 4.6.

4.2 Related literature and problem motivation

In a QDC, careful consideration must be given to the implementation of multi-qubit gates. In particular, however a quantum circuit is partitioned among QPUs, it is likely that there will be multi-qubit gates that act on qubits assigned to different QPUs. We refer to such inter-QPU gates as remote gates and refer to intra-QPU gates as local gates.

Most methodologies for carrying out remote gates boil down to one of two schemes. In the first scheme, which we call TP-comm (as in Ref. [21]), quantum teleportation [88, 89] is used to transfer quantum states from one QPU to another, and then the gate is conducted locally. Alternatively, as advocated for in Ref. [38], an entangled state can be maintained between the two QPUs while the local gate is applied. This latter idea is extended in Ref. [2], where it is shown that the remote gate methodology proposed in Ref. [38] can be decomposed into subprotocols [2], called cat-entanglement and cat-disentanglement, between which one or more local gates are conducted. In doing so, one can implement Shor's algorithm using fewer inter-QPU entangled pairs [3]—which are referred to as entangled bits or ebits. The term cat-comm [2, 3] is used here to describe implementing a remote gate or gates in this way.

Subsequent work [21, 90–100], compiles arbitrary quantum circuits into a form suitable for distributed quantum computing, with a view to minimising the number of ebits used. Many compiler proposals [21, 98, 100] involve a step we call communication assignment [21, 98, 100], in which cat-comm or TP-comm is assigned to remote gates or groups of remote gates. Current work on

communication assignment [21, 98, 100] compares cat-comm or TP-comm with respect to the number of ebits they each use or their latency cost. Ebits have a significantly larger error probability and latency than local gates, which is why they have been a primary consideration of previous work. However, a typical distributed quantum circuit will have far more local gates than ebits and so, prior to our work, there was no definitive evidence for prioritising ebit minimisation above all else. Latency considerations are also more complicated than they may initially appear due to the greater difficulty of correcting errors in quantum computers relative to classical ones. In current quantum computers, more noise means more runs of the circuit are required to obtain a given precision in the result, increasing the latency. Moreover, quantum error correction, which is expected to be incorporated into future quantum computers, only works if the errors are small [6]. Therefore, simply arguing that a process takes a long time relative to other processes does not automatically ensure that the overall latency will be lower. There are also many cases where cat-comm and TP-comm have the same ebit and/or latency cost for implementing a remote gate or group of remote gates [21], and so previous work gives no clear way of distinguishing between cat-comm and TP-comm in such circumstances.

In our work, we instead consider the output fidelity, F_{out}, after a remote gate or quantum circuit. F_{out} indicates the similarity between the ideal output, ρ_{ideal}, and the actual output, ρ_{noisy}, obtained in the presence of noise. Specifically [60], we use Eq. (2.5), which states

$$F_{\text{out}} = \left(\text{Tr} \sqrt{(\rho_{\text{ideal}})^{\frac{1}{2}} \rho_{\text{noisy}} (\rho_{\text{ideal}})^{\frac{1}{2}}} \right)^2 .$$

Crucially, this allows us to differentiate between cat-comm and TP-comm when the number of ebits is the same, and to understand the robustness of each scheme to different types of error. Using F_{out}, we can also compare the relative impact of different error types. With this far more detailed understanding of cat-comm, TP-comm, and errors, we hope to facilitate more optimal compilation heuristics in the future.

Our work is not the first to consider the errors in the QDC setting. Some of the earliest work to do so is Ref. [37]. However, the results of Ref. [37] are specific

to the quantum phase estimation algorithm [56, 101] and fewer types of noise are considered. The only local error considered is time-dependent dephasing, for which effective suppression techniques exist [102], and comparisons are conducted not between different types of noise, as we do, but between different implementations of the quantum phase algorithm—one in which QPUs can only communicate classically and another in which QPUs can also share quantum entanglement. Consequently, it may be difficult to ascertain from Ref. [37] where optimisation efforts for hardware and compilation should focus. Moreover, the figure of merit used by Ref. [37] is a cost function which treats the 'cost' of running a computation on each QPU and the 'cost' of sending measurement outcomes to a central QPU as abstract parameters and relates these 'costs' to the number of QPUs. This degree of abstraction makes the a priori linking of results to experiment challenging and implicitly assumes that intra-QPU errors are uncorrelated with the number of QPUs. In other words, they assume that the number of qubits and local gates on each QPU are fixed. This neglects the fact that cat-comm and TP-comm require additional local gates, rather than just ebits. By contrast, F_{out}, which we use, can be experimentally measured and has clear physical significance.

Later work [103] uses classical simulation to find the output fidelity of two different distributed implementations of the quantum phase estimation algorithms on a QDC. In one implementation, cat-entanglement and cat-disentanglement are conducted for each remote gate, whereas in the other implementation several remote gates are amalgamated by deferring cat-disentanglement until later in the circuit. The authors consider the impact of adding depolarising noise to inter-nodal entanglement and find that F_{out} declines less quickly with increasing depolarisation probability when the remote gates are amalgamated than when they are not. They also quantify the decrease in output fidelity with the number of devices for different depolarisation probabilities. However, no comparison between cat-comm and TP-comm is made and only entanglement error is considered.

A further limitation of Refs. [37, 103] is the specialisation to the quantum phase estimation algorithm. To generalise results, it is necessary to understand how the building blocks of circuits work, as well as how errors propagate through circuits with different structures. For this reason, we consider the impact of errors on individual remote gates, in this chapter, and, in the next chapter, we consider

the impact of errors on a wide variety of larger circuits. In this way, we are able to more deeply and generally understand the impacts of different errors and communication schemes than previous work.

4.3 Quantum data centres: main components

To understand where optimisation of future QDC manufacturing and compilation efforts should focus, it is important to understand the impact of different types of noise on a QDC. In this chapter, we quantify the extent to which the output of distributed quantum circuits is degraded by the different types of noise present in a QDC. The metric we use to do this is the output error, characterised by $1 - F_{\text{out}}$. We also compare the robustness of cat-comm and TP-comm to error.

For simplicity, we assume that just two QPUs are present in the QDC we consider. We assume that all remote gates in a given quantum circuit are distributed using the same communication scheme and compare the impact of different communication schemes on the output error. Further details on the schemes considered are provided in Sec. 4.3.1. We also consider different types of noise, which we model using the methods described in Sec. 4.3.2. Additional information about how entanglement is distributed between QPUs and how entanglement error could be reduced are discussed in Sec. 4.3.3.

4.3.1 Cat-comm and TP-comm circuits

Comparing cat-comm and TP-comm is not as simple as it may appear at first glance. In particular, there is some ambiguity in what it means to implement a remote gate using TP-comm. Therefore, in this work, we consider cat-comm and three different versions of TP-comm, which would be used in different scenarios within larger quantum circuits.

Consider first a remote CNOT gate implemented using cat-comm. The corresponding circuit diagram is shown in Fig. 4.1(a). The objective of this circuit is to implement a CNOT gate between QPU A's processing qubit, q_2, and QPU B's processing qubit, q_2'. q_2 is the control qubit and is in the arbitrary initial state $\alpha|0\rangle + \beta|1\rangle$, where α and β are complex numbers such that $|\alpha|^2 + |\beta|^2 = 1$. q_2'

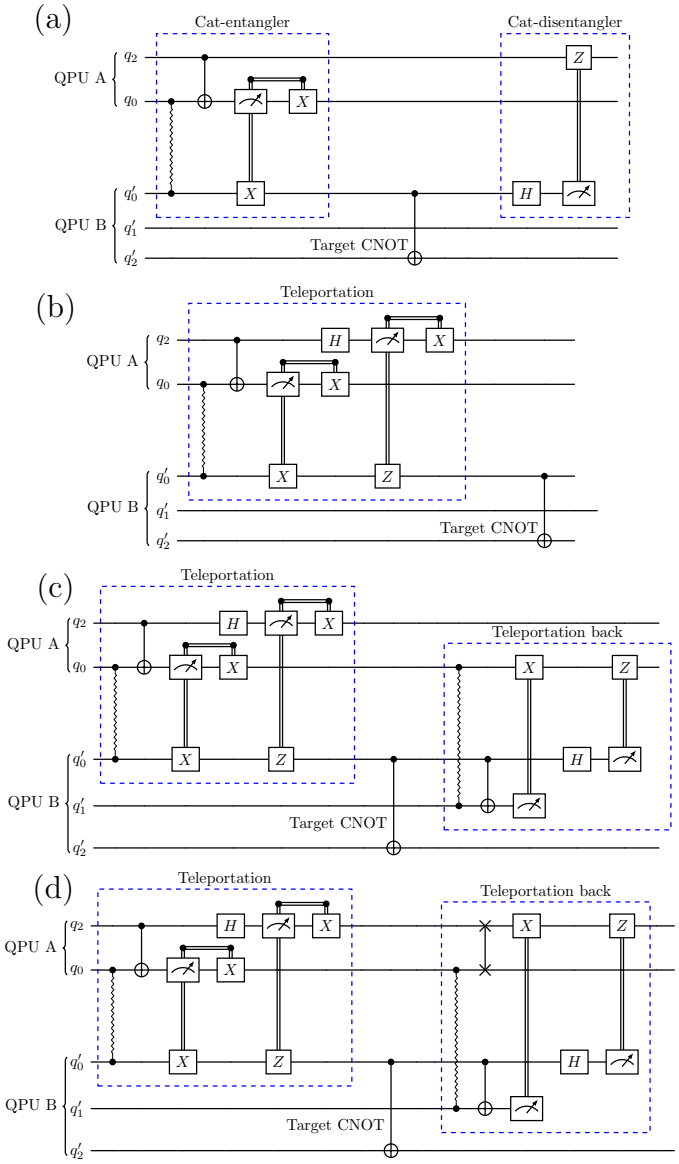


Figure 4.1: A remote CNOT gate implemented using: (a) cat-comm [2, 3]; (b) 1TP; (c) 2TP; and (d) TP-safe. Zigzags represent ebits, which here, in the ideal case, are Bell pairs in the state $|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$. Double lines represent classical communication. Gates classically connected to a measurement device activate on a measurement result of '1' only.

is the target qubit and is in the arbitary initial state $|\chi\rangle$. The communication qubits q_0, from QPU A, and q_0' from QPU B are also used.

The cat-comm remote CNOT gate consists of three subroutines: cat-entanglement, implementation of the CNOT locally, and then cat-disentanglement. Cat-entanglement entangles q_2 with q_0', so that they share the cat-like state, $\alpha|00\rangle + \beta|11\rangle$. In this way, the state of the control qubit, q_2, on QPU A, is shared with q_0', on QPU B. The CNOT gate can then be conducted locally between q_0' and q_2' before the entanglement between q_2 and q_0' is destroyed by the cat-disentanglement process shown in Fig. 4.1(a). A similar process can be used for any controlled-unitary (CU) gate.

Fig. 4.1(b) shows the circuit diagram for a CNOT gate implmented using our first variant of TP-comm, which we call 1TP. The same qubits that were used in cat-comm are again considered here. In 1TP, the state of q_2 is teleported to q_0' and then the CNOT gate is applied locally on QPU B. More generally, any local gate could be conducted after the teleportation. Unlike cat-comm, it is not necessary for the gate to be a CU gate.

The next TP-comm scheme, 2TP, is depicted in Fig. 4.1(c), again for the specific case of a remote CNOT. 2TP involves first applying 1TP and then teleporting the state of the control qubit back to QPU A. More specifically, the second teleportation is from q_0', on QPU B, to the communication qubit, q_0, on QPU A. An additional communication qubit is also needed on QPU B to facilitate the second teleportation.

Finally, a remote CNOT gate implemented using TP-safe, the final TP-comm scheme, is shown in Fig. 4.1(d). TP-safe is almost identical to 2TP except that the states of q_0 and q_2 are swapped during the second teleportation, so that the state of the control qubit is restored to where it first started: the processing qubit q_2. This is done using a SWAP gate.

It is not immediately obvious which of the TP-comm schemes discussed here is most comparable with cat-comm. On the one hand, cat-comm and 1TP both use exactly the same quantities and types of ebits, measurements, classical communications, and local gates. On the other hand, after cat-comm, q_0' is free to host any further ebits needed for additional remote gates. By contrast, after 1TP, q_0' is needed to store the teleported state of the control qubit. To avoid overwriting

this crucial quantum information when conducting future remote gates, a second teleportation can be used to free q_0'. As we consider only two QPUs, the second teleportation must be back to QPU A, as is done in 2TP. More generally, the teleportation could be to any other QPU. Teleportation back to the original QPU is the least favourable scenario, but it can often be necessary when large quantum circuits are compiled. After 2TP, q_0' on QPU B is left free, but q_0 on QPU A is occupied. To circumvent this issue, we can carry out a SWAP gate between q_0 and q_2 to restore the teleported state back to where it started prior to the remote gate. This is safe to do because we know that q_2 was measured and then re-initialised during the first teleportation and so carries no non-trivial quantum information.

Cat-comm and TP-safe can always be used, regardless of what subsequent remote gates are scheduled, because, at the end of either protocol, all communication qubits are free of non-trivial quantum information and can be re-used for further remote gates without overwriting anything. The same cannot be said for 1TP and 2TP, but they are less resource-intensive than TP-safe. In circumstances where no further remote gates are scheduled on the QPU that holds a teleported state, there is no need to free the communication qubits, and it will be better to use 1TP or 2TP than TP-safe. Otherwise, TP-safe is the only TP-comm scheme that could be used. TP-safe represents the very worst-case scenario that may be necessary during compilation of a distributed quantum circuit using TP-comm, and 1TP is the very best. By considering 1TP and TP-safe, we aim to provide an upper and lower bound on the performance of TP-comm, while 2TP sits somewhere between those two extremes.

4.3.2 Error models

Three principal sources of error are considered in this chapter: imperfections in the distributed ebits, imperfect implementation of local two-qubit gates, and time-dependent memory depolarisation. In each case, we model noise using depolarising channels, in which with some probability the ideal state that should be present is instead replaced by white noise. This type of noise modelling is used throughout this thesis for several reasons. Firstly, it is relatively standard across the field. More importantly, an arbitrary error channel can be reformulated into depolarising

noise by applying random single-qubit gates [7]. Finally, when it comes to trying to combat errors, as is considered in Chapter 6, depolarising noise is typically an unfavourable assumption because, if noise the noise is not uniform, and the deviation from uniformity is known, then more efficient schemes for combatting it can typically be inferred [104]. This means that depolarising noise can be viewed as something of a worst case scenario, making our results a lower bound on what is possible.

In cat-comm and TP-comm, we assume that, in the absence of noise, the ebits used have the pure state $|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$. When modelling imperfections in the ebits, we assume that their state is a Werner state of the form

$$\begin{aligned}\rho_{\mathrm{w}} =& F_{\mathrm{w}} \left|\Phi^+\right\rangle \left\langle\Phi^+\right| + \frac{1-F_{\mathrm{w}}}{3} \left(\left|\Phi^-\right\rangle \left\langle\Phi^-\right| \right. \\ &\left. + \left|\Psi^+\right\rangle \left\langle\Psi^+\right| + \left|\Psi^-\right\rangle \left\langle\Psi^-\right|\right),\end{aligned} \tag{4.1}$$

where $0 \leq F_{\mathrm{w}} \leq 1$ is a real number, and $|\Phi^{\pm}\rangle = \frac{1}{\sqrt{2}}(|00\rangle \pm |11\rangle)$, $|\Psi^{\pm}\rangle = \frac{1}{\sqrt{2}}(|01\rangle \pm |10\rangle)$ are the Bell states, which form a basis for all two-qubit entangled states. A pair of qubits which are collectively in a Bell state is referred to as a Bell pair. We refer to $\epsilon_{\mathrm{ebit}} = 1 - F_{\mathrm{w}}$ as the entanglement error.

In reality, the form of the non-ideal entangled states is unlikely to be exactly known and may not be so symmetrical in nature. Nonetheless, any mixture of bipartite entangled states can be transformed into the Werner state via a random bilateral rotation of each of the entangled qubits [7]. Bilateral rotations require only single-qubit gates with relatively low errors and are sometimes done as part of entanglement distillation [7, 64], which is likely to be necessary in the QDC context. Moreover, less symmetrical imperfections are often advantageous for error correction or mitigation [104]. As such, the Werner state is likely to represent a worst-case scenario, which is useful for benchmarking purposes.

Non-ideal application of local gates is modelled by applying a depolarisation channel of the form [105]

$$\rho_{\mathrm{in}} \to (1-\epsilon_{\mathrm{cnot}}) U_{i,j} \rho_{\mathrm{in}} U_{i,j}^{\dagger} + \frac{\epsilon_{\mathrm{cnot}}}{4} \mathrm{Tr}_{i,j}(\rho_{\mathrm{in}}) \otimes \mathbb{1}_{i,j}, \tag{4.2}$$

to all local two-qubit gates, which in this work are all CNOT gates. Here, ϵ_{cnot} is the error probability; $U_{i,j}$ is an ideal two-qubit gate operation on the ith and jth

qubits; ρ_{in} is the input density matrix; $\text{Tr}_{i,j}$ is the partial trace [56] with respect to the subspaces of qubits i and j; and $\mathbb{1}_{i,j}$ is the identity operation acting on the these subspaces.

Memory depolarisation on qubit k is modelled using the depolarisation channel

$$\rho_{\text{in}} \rightarrow e^{-\Delta t\, r}\rho_{\text{in}} + \frac{(1 - e^{-\Delta t\, r})}{2}\text{Tr}_k(\rho_{\text{in}}) \otimes \mathbb{1}_k, \tag{4.3}$$

where Δt is the time since qubit k was initialised, r is the hardware dependent depolarisation rate, and ρ_{in} is the input density matrix.

4.3.3 Entanglement distribution and distillation

Entanglement distribution can be carried out in several ways. For instance, photons can be entangled with communication qubits and then measured out, leaving the communication qubits entangled with one another. The measurement process used is typically a Bell state measurement (BSM) [106]. A BSM interacts and measures a pair of qubits to ascertain whether those qubits are in one of the Bell states. In principle, a BSM can be conducted using a CNOT gate followed by a hadamard gate on the control qubit [106]. This is known as a deterministic BSM. In practice, deterministic BSMs may be difficult to achieve and so partial BSMs using linear optics and photodetectors are frequently used. A partial BSM relies on post-selection of measurement results to probabilistically determine whether a pair of qubits are in one of the Bell states.

Regardless of whether a partial or deterministic BSM is used, most entanglement distribution schemes require the entire entanglement distribution process to be repeated several times. For partial BSMs, this is an intrinsic feature of their probabilistic nature. For deterministic BSMs, repetition is necessitated by the presence of loss in the communication channel. If photons do not reach the detectors in a BSM module, new photons must again be entangled with communication qubits, and sent until they reach the detectors. In this way, loss is heralded by the lack of proper detection events.

The fidelity of the ebits can also be improved by using a more advanced heralding process and imposing more stringent fidelity requirements on the entangled

states distributed [106]. This will lower the success rate of entanglement, necessitating a greater number of entanglement distribution attempts. Alternatively, but similarly, entanglement distillation [7, 8, 64, 107] could be used. In entanglement distillation, multiple entangled pairs are distributed between QPUs and local gates are conducted between communication qubits, entangling them. Some of the qubits are then measured out and inferences are made about the fidelity of the remaining qubits. This process is repeated iteratively until the fidelity reaches the desired threshold.

Other options for improving the fidelity of ebits include applying quantum error correction [6] locally on the communication qubits for each QPU and encoding the ebit into a larger many-qubit entangled (cluster) state [106].

Regardless of how loss and noise are mitigated, there is a cost to be paid. All of the aforementioned schemes for improving the fidelity of ebits involve a latency cost due to repetitions of the entanglement process, decoding time, or the addition of local gates. Any additional local gates will also be imperfect, adding further noise into the system. Moreover, in many cases, redundant communication qubits and/or photons are also required. Therefore, there is always a trade-off to be made between improving ebit fidelity and paying the price of doing so.

In this work, we abstract from the details of the entanglement distribution scheme used by assuming that communication qubits are entangled at a fixed rate R. The time taken for a given ebit to be distributed is assumed to always be $\frac{1}{R}$ from when the ebit is requested during the compilation process. Ebit requests are generated when the ebit is first needed, which is unfavourable for latency. That said, it does minimise the time that ebits have to decohere after they have been distributed. With these assumptions, all loss is incorporated into the value of R rather than being explicitly modelled. Noise is accounted for by assuming that the ebit is in the Werner state as discussed in Sec. 4.3.2.

Despite the abstractions we make, it is important to keep in mind the trade-offs inherent to altering R or reducing the entanglement error. Even with current technology, entanglement error could be reduced from the current state-of-the-art values but, as discussed previously, this would come at the expense of a lower entanglement distribution rate and often requires additional qubits and local gates. These trade-offs are important to keep in mind when contemplating our results.

4.4 Error analysis

4.4.1 Analytical approach

A natural starting point for any error analysis is to derive analytical results where possible. Even analytical results related to a small subset of the broader problem can be used to verify the results obtained by other methods. Here, we present closed form analytical results for arbitrary remote controlled-unitary gates implemented using 1TP and cat-comm in the presence of entanglement error only. All other errors are assumed to be zero. We also make the limiting assumption that the input state to the qubits is initially separable. As such, the input state to the remote controlled-unitary gate can be represented as

$$(\alpha\,|0\rangle + \beta\,|1\rangle)_{q_2} \otimes |\chi\rangle_{q_2'}\,, \tag{4.4}$$

where the subscripts q_2 and q_2' refer to the qubits shown in Fig. 4.1(b), α and β are complex numbers such that $|\alpha|^2 + |\beta|^2 = 1$, and $|\chi\rangle$ is an arbitrary pure state. We defer the derivation to Appendix A for brevity.

For 1TP, we find that the output fidelity, F, varies linearly with the Werner state fidelity F_{w} according to the equation

$$F_{\mathrm{out}} = \frac{1 + 2F_{\mathrm{w}}}{3}. \tag{4.5}$$

Remarkably, this expression is independent of which controlled-unitary operation is applied or the input to that controlled-unitary operation. This means that, when only ebit errors are present, a 1TP remote gate can always be performed with the same accuracy on any state and for any gate.

The situation is more complicated for cat-comm. We find that, when only ebit errors are present, the output fidelity of an arbitrary remote gate, implemented using cat-comm, depends on the Werner state fidelity, the input state and the specific gate implemented, according to the equation

$$\begin{aligned} F_{\mathrm{out}} = F_{\mathrm{w}} + \frac{1 - F_{\mathrm{w}}}{3}\Bigg(& \left(|\alpha|^2 - |\beta|^2\right)^2 + 2|\alpha|^4 \left|\langle\chi|\, U_{q_2'}\, |\chi\rangle_{q_2'}\right|^2 \\ & + 2|\beta|^4 \left(\langle\chi|\, U_{q_2'}^{\dagger}\, |\chi\rangle_{q_2'}\right)^2\Bigg), \end{aligned} \tag{4.6}$$

where $U_{q_2'}$ is the unitary operation applied to q_2' during the remote CU gate. As with Eq. (4.5), this depends linearly on F_{w}, but, unlike Eq. (4.5), Eq. (4.6) does depend on the type of CU gate applied and the input state of the control and target qubits.

Considering the case where a CNOT gate is applied and adding the further limiting assumption that $|\chi\rangle$ is an eigenstate of the computational basis, Eq. (4.6) simplifies to

$$F_{\mathrm{out}} = F_{\mathrm{w}} + \frac{1 - F_{\mathrm{w}}}{3}\left(2|\alpha|^2 - 1\right)^2. \tag{4.7}$$

Although, it applies to a very specific situation, Eq. (4.7) is convenient for testing other error analysis tools. The simpler form and reduced number of variables compared with Eq. (4.6) make Eq. (4.7) easier to plot and compare with the results obtained using other methods.

4.4.2 First-order approximations

To begin extending beyond the limiting assumptions of the analytical analysis, it is instructive to make first-order approximations to the results. These first-order approximations carry assumptions of their own but allow local multi-qubit gate errors to also be considered. Here, and throughout the remainder of this work, we use the U_3 + CNOT set of gates and so the only multi-qubit gates considered are CNOTs.

For sufficiently small local gate and entanglement error rates, ϵ_{cnot} and ϵ_{ebit}, respectively, one might expect the output fidelity, F_{out}, to degrade linearly with the number of imperfect CNOT gates, n_{cnot}, or ebits, n_{ebit}, yielding the equation:

$$F_{\mathrm{out}} \approx 1 - n_{\mathrm{cnot}}\epsilon_{\mathrm{cnot}} - n_{\mathrm{ebit}}\epsilon_{\mathrm{ebit}}. \tag{4.8}$$

Alternatively, retaining the main desired terms in Eqs. (4.1) and (4.2), we can approximate F_{out} by

$$F_{\mathrm{out}} \approx (1 - \epsilon_{\mathrm{ebit}})^{n_{\mathrm{ebit}}}(1 - \epsilon_{\mathrm{cnot}})^{n_{\mathrm{cnot}}}. \tag{4.9}$$

This is consistent with Eq. (4.8) in the limit of $n_{\mathrm{ebit}}\epsilon_{\mathrm{ebit}}$ and $n_{\mathrm{cnot}}\epsilon_{\mathrm{cnot}} \ll 1$ and has been used to estimate the fidelity in the presence of gate errors for larger circuits [108].

Both Eqs. (4.8) and (4.9) are quite general and can be used to describe local gate errors and entanglement errors for any quantum circuit that adheres to the assumptions used during derivation.

Memory depolarisation is continuous rather than discrete like local gate errors and entanglement errors. For this reason, memory depolarisation is not well described by an equation with the same form as Eq. (4.8) or Eq. (4.9), and we do not consider a first-order approximation for memory depolarisation in this work.

4.4.3 Simulation

In order to move past first-order approximations and enable extension to larger circuits, we generate numerical results using classical simulation. The simulator used [76] is an event-based simulator built using Python 3.9 and the associated libraries: NetSquid [66] and nuqasm2 [77]. We modified nuqasm2 for our purposes.

Our simulation package converts arbitrary monolithic quantum circuits, specified using openQASM 2 [82] or as a list of gate tuples, to compiled distributed quantum circuits. The use of communication qubits is handled automatically without explicit user specification and the subroutines associated with each type of remote gate specified in Fig. 4.1 can be automatically generated and scheduled. Manual specification of subroutines is also possible. All simulated hardware be associated with a corresponding noise model, and simulated QPUs come with a variety of options, including deciding the size of the communication and processing qubit allocations. Efforts have been made to retain the modularity of the NetSquid package and users can also specify their own compilers, QPUs and connections for use with our simulation package. Further details of our simulation package can be found in Chapter 3.

As the simulator is event-based, anything that happens during the simulation is allocated an amount of time it will take to occur. Time is discretised based on the duration of those events. After each event, the memory depolarisation channel, Eq. (4.3), is applied with Δt redefined to be the time since the last simulation event occurred. More concretely, in our simulations, memory depolarisation is applied before and after each remote gate, but the time taken for any proceeding local gates or classical simulation is accounted for in Δt. For simplicity, the classical

communication latency is modelled using NetSquid's fibre delay model, which assumes classical communication occurs in $\frac{d}{2\times10^8 \mathrm{ms}^{-1}}$, where d is the inter-QPU distance in metres. Entanglement distribution is carried out using the abstract model described in Sec. 4.3.3.

4.5 Numerical results

Armed with the tools described in Sec. 4.4, we are able to obtain numerical values for the output error. All simulated results are obtained using either a laptop with 16 GB of RAM and an AMD Ryzen 7 4700 processor or a desktop with 32 GB of RAM and an i7 processor.

We assume the simulated QDC has two QPUs each possessing just two communication qubits, which is believed to be a realistic limitation for near-term devices [21, 98]. We also assume that the processing qubits have an input state of

$$\frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right)_{\mathrm{q}_2} \otimes |0\rangle_{\mathrm{q}_2'}\,, \tag{4.10}$$

where the subscripts q_2 and q_2' refer to the corresponding qubits in Fig. 4.1. The control state was found to be most susceptible to entanglement errors out of all separable input states that have q_2' in the state $|0\rangle_{\mathrm{q}_2'}$, as discussed in Appendix B.

The parameter values used for the different types of error and the durations of different operations are based on state-of-the-art values for trapped-ion systems from the literature, and are displayed in Table 4.1. Where possible, parameters for current commercial hardware (IonQ's Aria quantum computer [110]) are used to inform the state-of-the-art-values, so as to make the simulated set-up more realistically implementable in the near-term. The parameter T_1 in Table 4.1 refers to the qubit lifetime, which informs the memory depolarisation rate, r. r is given by $r = \frac{1}{T_1}$. When a single state-of-the-art value of r is required we use the midpoint, $r = 0.055$, of the quoted range.

For the other parameters, when we consider a range of parameter values, we refer to the order of magnitude in which the relevant state-of-the-art parameter exists as the state-of-the-art range. To facilitate comparison between entanglement error and local gate error, we also often consider what happens when both error

Table 4.1: The nominal state-of-the-art parameter values used in Chapters 4 and 5.

Parameter	Value	Source
F_{w}	0.94	With $^{88}Sr^{+}$ qubits [109]
ϵ_{cnot}	0.4%	With $^{1}71Yb^{+}$ qubits [110]
Single-qubit gate time	135μs	[110]
Two-qubit gate time	600μs	[110]
Measurement time	6 ms	Inferred from [21, 110, 111] [1]
R	182 Hz	[109]
T_1	10 – 100s	[110]
d	2m	[109]

types are varied over the same range. When we do this, we consider a range of entanglement error which is an order of magnitude smaller than the 'state-of-the-art' value. As this entanglement error reduction would require one of the methods from Sec. 4.3.3, we refer to the resulting range as the distilled range, in reference to entanglement distillation, although entanglement distillation is just one of the possible error reduction methods that could be used.

In the following, we first compare our different error analysis tools. This is done in Sec. 4.5.1 for the building block modules shown in Fig. 4.1. After that, we compare cat-comm, 1TP, 2TP, and TP-safe in Sec. 4.5.2. Finally, we compare the impact of different types of error in Sec. 4.5.3.

4.5.1 Comparison of error analysis methods

As alluded to in Sec. 4.4.1, analytical results provide a useful robustness check for simulated ones. In our case, we find that the analytical expressions, Eq. (4.5) and Eq. (4.6), agree exactly with the simulated results for entanglement error, which suggests that the simulated results are likely to be robust.

The agreement is less strong between the analytical/simulation results and the first-order approximations given by Eqs. (4.8) and (4.9). Indeed, we find both first-order expressions to be inadequate, indicating the need for more detailed error analysis, such as the classical simulation of errors conducted in this work.

The failure of the linear approximation is well established for larger monolithic circuits [112], however, here we indicate that clear discrepancies between both first-order approximations and the simulated results can occur even for individual remote gates, when error parameters within the state-of-the-art or distilled ranges are used.

To demonstrate the inadequacy of the first-order approximation, we consider the percentage difference between simulated and approximate output errors, Δ_{oe}. This is calculated using the expression:

$$\Delta_{\text{oe}} = \frac{(1-(F_{\text{out}})_{\text{approx}}) - (1-(F_{\text{out}})_{\text{sim}})}{1-(F_{\text{out}})_{\text{sim}}} \times 100. \tag{4.11}$$

A positive sign to the result indicates that the first-order approximation overestimates the output error, $1-F_{\text{out}}$, relative to the simulated data, and a negative sign means that the first-order approximation underestimates the output error. The subscript 'sim' refers to simulated data and 'approx' to data calculated using Eq. (4.8) or Eq. (4.9).

Figure 4.2 shows Δ_{oe} after a remote CNOT gate is implemented using each of the schemes in Fig. 4.1. A local CNOT gate conducted on a single monolithic quantum computer is also considered for comparison. Figures 4.2(a) and 4.2(b) show the Δ_{oe} with respect to the entanglement error, varied over the state-of-the-art and distilled ranges, respectively. All other errors are set to zero. Figure 4.2(c) shows Δ_{oe} with respect to the local two-qubit gate error. Again all other errors, including entanglement errors, are set to zero.

Using Fig. 4.2, several observations can be made:

i. The first-order approximations provide an upper bound on the output error. All Δ_{oe} values are non-negative in Fig. 4.2. This means that the output errors obtained using the simulation never exceed the approximate ones. This observation is resilient to the choice of input state as discussed in Appendix B.2.1.

ii. The first-order approximations fail to distinguish between cat-comm and 1TP, for which the number of gates, classical communications, measurements, and ebits is the same. By contrast, the markedly different results for cat-comm and 1TP in Fig. 4.2 indicate that such a discrepancy does exist. This is one case where

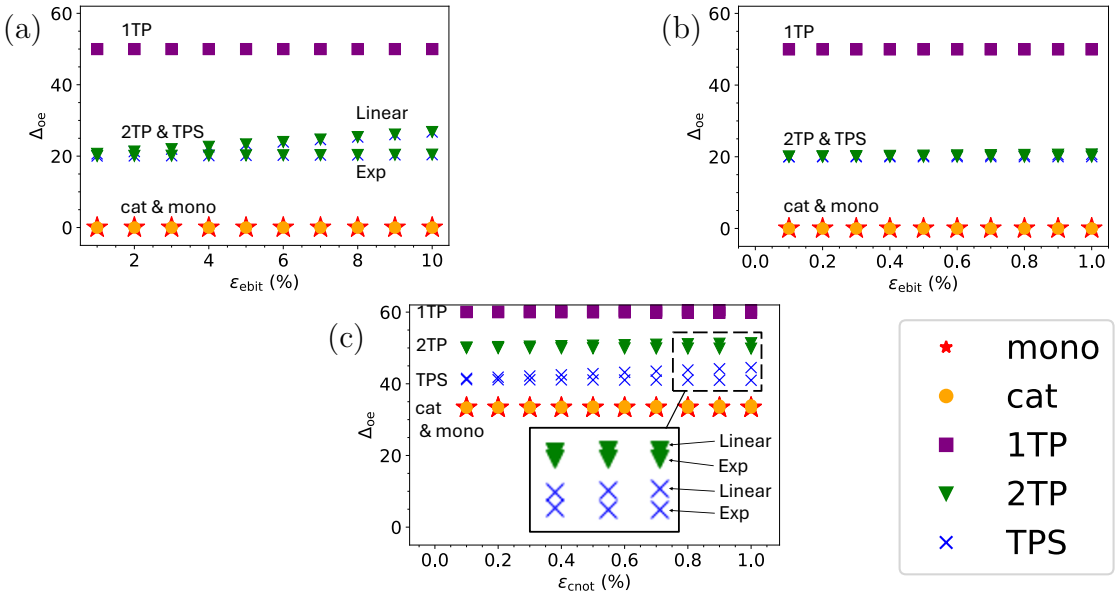


Figure 4.2: The percentage difference, Δ_{oe}, between the simulated and approximate results for a remote CNOT gate conducted using each of the schemes depicted in Fig. 4.1: 1TP, cat-comm (cat), 2TP, and TP-safe (TPS). The results for a local CNOT gate conducted on a monolithic processor is also shown for comparison (labelled 'mono'). In each case, the percentage difference, calculated using Eq. (4.11), is plotted with respect to: (a) the entanglement error, ϵ_{ebit}, varied over state-of-the-art range, with all other errors set to zero; (b) the entanglement error, ϵ_{ebit} varied over the distilled range, with all other errors set to zero; and (c) the local CNOT depolarisation errors, ϵ_{cnot}, varied over the state-of-the-art-range, with all other errors set to zero. The approximate results are calculated using Eqs. (4.8) and (4.9), marked as 'Linear' and 'Exp', respectively, when a difference is discernible.

the numerical simulation reveals behaviour that is not apparent from first-order analysis alone.

iii. The discrepancy between simulated and approximate results can be large. For the input state chosen, this discrepancy is especially significant for 1TP, 2TP, and TP-safe. Figures 4.2(a) and 4.2(b) indicate that the discrepancy is approximately 20-50% when only entanglement error is considered and Fig. 4.2(c) indicates that it is 40-60% when local two-qubit gate errors are considered. Therefore, even with just one remote gate, the difference in the output error predicted by the first-order approximations and the simulation can be significant.

iv. Good agreement between simulated and first-order results is possible for some schemes. For cat-comm and the monolithic case, the simulated and first-order results agree when only entanglement error is considered. This can be seen from Figs. 4.2(a) and 4.2(b). This agreement is highly dependent on input state as discussed in Appendix B.2.1.

v. The linear and exponential approximations, given by Eqs. (4.8) and (4.9), respectively, perform similarly over the investigated error ranges, but, even with just one remote gate, the exponential approximation can be seen to perform slightly better than the linear one for 2TP and TP-safe. This holds for both entanglement error and local two-qubit gate error, within the state-of-the-art range, as can be seen from Figs. 4.2(a) and 4.2(c).

A key takeaway from these observations is that we cannot rely on first-order approximations alone for a quantitative error analysis of a QDC. Given the complexity of closed-form analytical analysis of arbitrary distributed quantum circuits, numerical simulation is therefore a key tool for understanding the role of errors in the QDC context.

4.5.2 Comparison of remote gate implementations

Using numerical simulation, we are able to compare different ways of implementing remote gates. The remote gate varieties we consider are shown in Fig. 4.1. The distributed quantum computing literature [2, 3, 21, 38, 90–92, 94–99, 113–115] typically bases such comparisons on the number of EPR pairs required by each

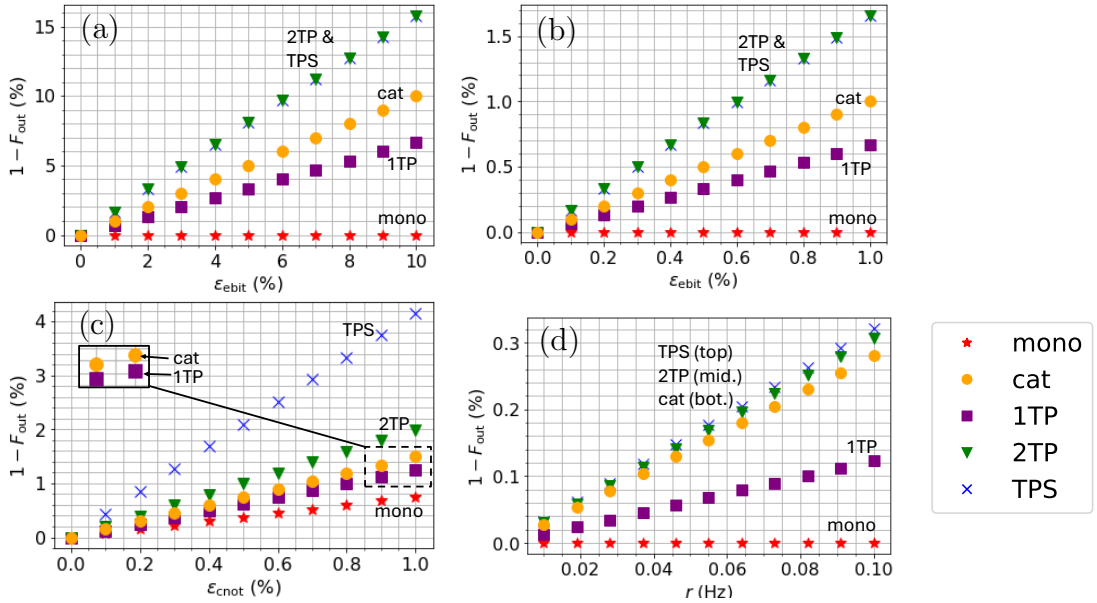


Figure 4.3: The output error, $1 - F_{\text{out}}$, for an individual remote CNOT gate with increasing: (a) entanglement error, ϵ_{ebit}, (state-of-the-art range); (b) entanglement error, ϵ_{ebit}, (distilled range); (c) local two-qubit gate error, ϵ_{cnot}; and (d) memory depolarisation, r, for cat-comm (cat), 1TP, 2TP, and TP-safe (TPS). For the monolithic case (mono) a single local CNOT gate is considered. The input state is given by Eq. (4.10).

scheme in various contexts, which falls short of the more detailed analysis possible using numerical simulation.

Figure 4.3 shows the output error, $1 - F_{\text{out}}$, after a remote CNOT gate. We vary one error at a time, respectively varying: entanglement error, over the state-of-the-art range in Fig. 4.3(a) and the distilled range in Fig. 4.3(b); local two-qubit gate error in Fig. 4.3(c); and memory depolarisation rate in Fig. 4.3(d). Again, all error parameters not explicitly varied in a given subfigure are set to zero.

Two key observations can be made:

i. For the input state given by Eq. (4.10), the remote gate schemes are ordered from lowest to highest by their impact on output error, as: 1TP, cat-comm, 2TP, and then TP-safe, where 2TP and TP-safe are equally damaging to the output when only entanglement error is considered. This ordering leads to the next more general observation.

ii. Cat-comm and 1TP can be distinguishable. This is consistent with observation ii. from Sec. 4.5.1, but is easier to see using Fig. 4.3. As discussed previously, it is not obvious a priori that cat-comm and 1TP could lead to different output errors, as they use exactly the same gates, measurements, ebits and classical communication. The difference between the output error produced by 1TP and cat-comm may offer an optimisation opportunity if either scheme is found to perform better when averaged over all input states. Such averaging is beyond the scope of this chapter and we defer averaged comparisons of cat-comm and 1TP to future work.

4.5.3 Comparison of error types

Although the relative qualitative performance of different schemes is unchanged by the type of error considered, another important algorithmic consideration for any quantum computer is how best to mitigate or, ideally, correct errors. A greater understanding of which errors have the most impact may help focus such efforts on where they are most needed and assist the efficient deployment of hardware resources to get the best possible output with the least possible effort. To this end, we investigate the impact of different types of error on the output error.

We gain insight into the relative impact of different error types in two different ways. Firstly, we compare the quantitative values of output error displayed in the different subfigures of Fig. 4.3. This allows us to re-use Fig. 4.3 to gain insight into the relative impact of errors when entanglement error is varied within the state-of-the-art range. However, this analysis is insufficiently precise to compare errors when entanglement error is varied within the distilled range, where the magnitudes of output error are much smaller. To perform comparison between entanglement error in the distilled range and other types of error, it is useful to directly plot the output error caused by each type of error on the same figure. For this reason, in Fig. 4.4, we collect the output fidelity data used in Figs. 4.3(b)-(d), and re-express it all on one plot per remote gate scheme. In each plot, the loss of output fidelity due to entanglement error only (EEO), two-qubit gate error only (GEO), and memory depolarisation only (MDO) are all shown.

Using Figs. 4.3 and 4.4, we find that:

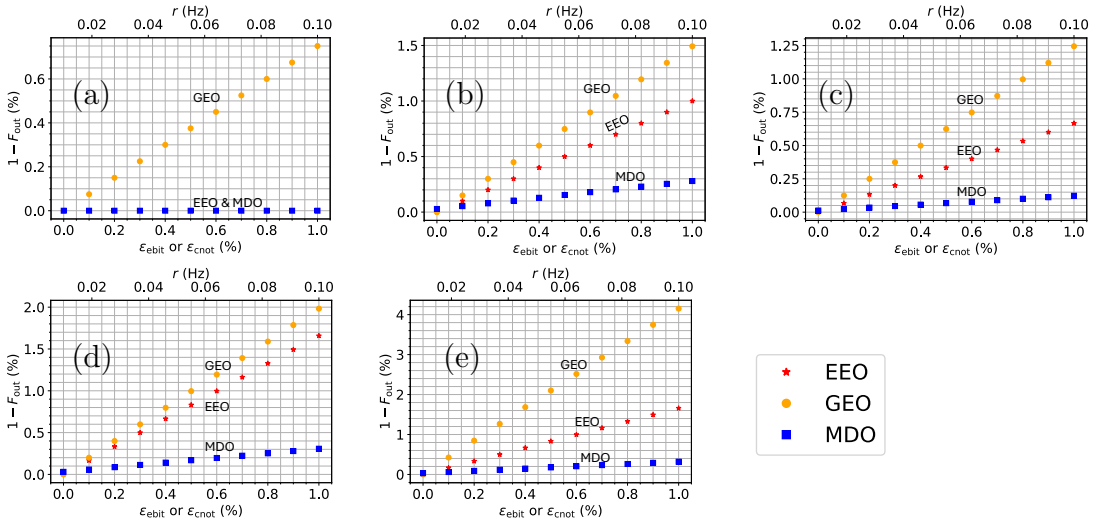


Figure 4.4: The output error, $1 - F_{\text{out}}$ for a single remote (local for monolithic case) CNOT gate, implemented using: (a) the monolithic case, (b) cat-comm, (c) 1TP, (d) 2TP, (e) TP-safe. The input state is given by Eq. (4.10). Each set of markers corresponds to the results obtained by varying a single error parameter and setting all other errors to zero. EEO indicates entanglement error only, GEO indicates gate error only, and MDO indicates memory depolarisation only. As the units of r are different from those of ϵ_{ebit} and ϵ_{cnot}, two different horizontal axes are used, to facilitate comparison between the impact of the different error parameters over an experimentally meaningful range. The bottom horizontal axis is used by the EEO and GEO curves to show the range over which the entanglement error rate, ϵ_{ebit}, and the local two-qubit gate error rate, ϵ_{cnot}, respectively, vary. This means that the entanglement error is reduced by an order of magnitude relative to the state-of-the-art range, to facilitate comparison with the two-qubit gate error. The MDO curve is governed by the top horizontal axis, which shows variation in the memory depolarisation rate over the state-of-the-art range.

i. When entanglement error is varied over the state-of-the-art range, in our simulation, the error types are ranked from least to most detrimental to output error as: memory depolarisation, local two-qubit gate error, and then entanglement error.

Memory depolarisation is relatively negligible. When only memory depolarisation is non-zero, as in the bottom curve of Figs. 4.4(a)-(e), we can see that memory depolarisation leads to significantly less output error than local CNOT error for all considered depolarisation rates.

Local two-qubit gate error is in turn dominated by entanglement error. This can be seen by comparing Fig. 4.3(a), in which only entanglement error is non-zero,

with Fig. 4.3(c), in which only local two-qubit gate error is non-zero. In Fig. 4.3(c), only TP-safe ever has an output error exceeding 2% for any local gate error in the considered range and there is no data point where the output error exceeds 4.2%. In Fig. 4.3(a), all schemes have an output error exceeding 2% as soon as the entanglement error reaches 4% and at 8% entanglement error, the output error of all schemes is greater than 4.2%.

ii. When entanglement error is varied over the distilled range, in our simulation, the ranking of the error types from least to most detrimental to output error changes to: memory depolarisation, entanglement error, and then local two-qubit gate error.

The ordering of entanglement error and local two-qubit gate error flips now that their magnitude is varied over the same range. This is easily seen from Figs. 4.4(a)-(e). This finding is not immediately obvious a priori, as one might expect the correlations intrinsic to entanglement to propagate errors more rapidly through the system.

Again, memory depolarisation is negligible and its impact is significantly lower than the other types of error shown in 4.4(a)-(e) for the parameter values considered here.

Overall, the findings suggest that entanglement error will be the most urgent optimisation consideration in the near term, however if the entanglement error is reduced, for example by using one of the methods from Sec. 4.3.3, the local gate errors may well be more significant. The trade-off between improving entanglement errors and introducing additional local errors and/or latency via the methods discussed in Sec. 4.3.3 would be an interesting and potentially very important point of future study.

4.6 Conclusion

In this chapter, we used classical simulation to emulate operations on a quantum data centre. We studied the behaviour of individual remote gates in the presence of various types of error. We found that first-order approximations to error propagation fail to accurately capture the behaviour of even the smallest

systems for any finite error. We also found that for error values obtainable with current trapped-ion hardware, the detrimental impact of imperfect inter-nodal entanglement dominates that of local errors, due to the much higher magnitude of entanglement error. The next most impactful of the errors considered is imperfect implementation of local gates, and time-dependent memory depolarisation is negligible in comparison to other forms of error. However, if high-quality entanglement could be generated, so that the magnitudes of the gate and entanglement errors are comparable, then gate errors become more impactful than entanglement errors. Finally, we discovered that, despite having the same quantities and types of ebits, gates, classical communications, and measurements, cat-comm and 1TP can have different output errors for the same error types and parameter values. The exact discrepancy in output error is input state dependent.

In the next chapter, we extend this study to larger circuits to better understand the impact of error in QDCs.

Chapter 5

Noise analysis of larger quantum circuits

5.1 Introduction

The behaviour of individual remote gates can tell us many things, however, when many such gates operate in tandem, any errors in one gate will impact the rest of the system. How this happens may depend on the structure of the circuits in question and so it is not automatically obvious that the trends seen for individual remote gates will also hold for larger circuits. In this chapter, we verify that the observations on the relative impact of different errors made in the previous chapter do in fact hold for larger quantum circuits and discuss some differences between single remote gates and larger circuits.

Our main contributions are:

- We determine the relative impact of noise in inter-QPU entanglement, remote CNOT gates, and decohering qubits (as time evolves) on the performance of a wide variety of distributed quantum circuits implemented on a QDC.
- We investigate the propagation of errors as the size of the QDC is increased.
- We infer several observations on the feasibility of near-term QDCs, implemented using matter-based qubits with photonic interconnects.

In Sec. 5.2, we introduce the circuits considered in this chapter before discussing how they are compiled in Sec. 5.3. Sec. 5.4 follows on from this with a discussion of the results obtained. Finally, the results and observations are summarised in Sec. 5.5.

5.2 Circuits considered

In order to get an accurate picture of the ways in which noise affects QDCs it is important to consider a variety of applications so as to ensure that the results are not specific to certain types of algorithm. To this end, in this chapter, we consider 22 different types of quantum circuits taken from the MQT bench library [4].

The circuits considered correspond to the following types of algorithm:

1. Amplitude estimation (AE)
2. The Deutsch-Jozsa (DJ) algorithm
3. GHZ state generation
4. Graph state generation
5. Grover's algorithm
6. The Quantum approximation optimisation algorithm (QAOA)
7. Quantum Fourier transformation (QFT)
8. Quantum neural network
9. Quantum phase estimation (QPE)
10. Quantum walk
11. Random circuit
12. Variational quantum eigensolver (VQE)
13. Three VQE ansatz functions referred to as two local, real amplitudes and efficient SU2

14. W-state preparation

Multiple variants of the Grover's algorithm, the quantum walk, phase estimation, and VQE are considered, and both QAOA and VQE are applied to several use cases, bringing the total number of circuits to the previously stated 22. This excludes implementations of scaled versions of the same circuit with a different number of qubits used. We predominantly consider the five-qubit implementations of these algorithms but do also consider the impact of varying qubit number on each algorithm.

The algorithms used represent a variety of applications. Many quantum algorithms, including some of those believed to be the most useful in the fault-tolerant regime of quantum computing, are underpinned by QFT and quantum walks [116]. One such algorithm, QPE, is also commonly used as a subroutine [56] and we consider it separately. QPE is foundational to the famous Shor's algorithm [10] for factoring, as well as the historically important but less useful DJ algorithm, which we also consider.

Quantum walks are the quantum version of a classical walk and offer speed ups on a variety of Markov chain problems [116], which are ubiquitous in statistical modelling of real-world problems. Quantum walks can be related to significant search and sampling algorithms including Grover's algorithm, in the sense that Grover's algorithm can be expressed as a specific quantum walk [117], although, in our case the specific quantum walk that we consider does not implement either algorithm explicitly and we consider all Grover's algorithm and the quantum walk separately. Grover's algorithm offers quadratic speed up of an unordered data base search [11], which is of great practical usefulness. Within the domain of search and sampling algorithms, we also consider amplitude estimation, which is used to estimate the amplitude of a specific qubit within a large, mult-qubit system and offers a quadratic speed up for a variety of problems that would conventionally be solved using classical Monte Carlo simulation such as risk analysis, option pricing and numerical integration [118].

For nearer term applications, the NISQ hybrid algorithms QAOA and VQE are considered, along with three ansatzes for the latter. The GHZ, graph and W states, whose generation circuits we considered, are also potentially useful for a

variety of applications in quantum communications and sensing protocols, which may not require fault-tolerant hardware. All three states also represent different forms of multipartite entanglement making them a useful benchmark.

Finally, the quantum neural network adds a machine learning specific application to the circuits considered and the random circuit, which is twice as deep as wide and considers entirely random gates acting on four qubits or fewer, has no practical application but is included as a further, less consciously structured benchmark.

Collectively, the circuits considered have relevance to long and near-term use cases, practical settings, quantum communications and sensing, and historically relevant proposals. Results are taken individually for each of the circuits considered and we find that the over-arching qualitative findings of this chapter hold true in every case. As such the findings seem to be relatively general.

5.3 Compilation

To maintain this generality, the choice of compilation is important. It would be easy to incorporate a bias into the structure of circuits through the compilation choices. For this reason, we use the lightest touch possible during compilations, leaving the initial structure of the circuits intact as much as we can. MQT bench offers multiple levels of abstraction in its circuit specifications for a given algorithm and can utilise the Qiskit compiler produced by IBM and the TKET compiler produced by Quantinuum. We use MQT Bench's target-independent level specification which produces circuits independent of the target architecture, to keep things as generic as possible.

Once the monolithic circuit is produced, compilation into a distributed circuit is kept simple. All quantum circuits are distributed by simple bipartitioning between two QPUs. Half of the processing qubits are given to one QPU and half to the other. If there is an odd number of processing qubits, then the additional qubit is arbitrarily allocated to a QPU based on the indices supplied for the original monolithic circuits by MQT bench [4]—the QPU whose qubits have the lower indices are given the extra qubit. As in the previous chapter, we assume

that there are two communication qubits per QPU. Scheduling of operations is done greedily, but, for simplicity, ebits are requested only when needed.

No attempt is made to merge remote gates together or otherwise optimise compilation. In this way, we avoid making the results specific to any one distributed quantum computing compiler, which may have imposed unforeseen structural bias on the circuits. Therefore, results represent a lower bound on what can be achieved with QDC, and further optimisation is possible if the hardware is specified in greater architecturally specific detail and compilation is adjusted to optimise the circuits for that hardware.

5.4 Numerical results

We use the noise models described in Sec. 4.3.2 of the previous chapter and the numerical simulation methods described in Sec. 4.4.3.

The reasons for considering multiple implementations for TP-comm become apparent for these larger quantum circuits, as it becomes impossible to implement 1TP in many cases and even 2TP is often inapplicable with the simple compilation strategy employed. Therefore, we only consider TP-safe and cat-comm, which are compatible with the simple compilation strategy used. For each circuit considered, results are taken for when all remote gates are conducted with cat-comm and independently when all remote gates are conducted with TP-safe.

It is convenient to concisely express the behaviour of all 22 of the circuits investigated in one place. Doing this also allows the very different quantitative values obtained for different circuits to be directly compared. To this end, in Fig. 5.1, we consider the output error as a function of the number of remote gates. When multiple circuits share the same number of remote gates, all data points are shown—one for each circuit. Once again, unless otherwise noted, each type of error discussed in Sec. 4.3.2 is considered separately, with the other errors set to zero. When a given error is being considered, the relevant error parameter is set to the state-of-the-art value stated in Table 4.1. Here, we also include an additional curve corresponding to the output error when all of the errors are set to their state-of-the-art value at once.

Figure 5.2 shows the same thing as Fig. 5.1 except that the entanglement

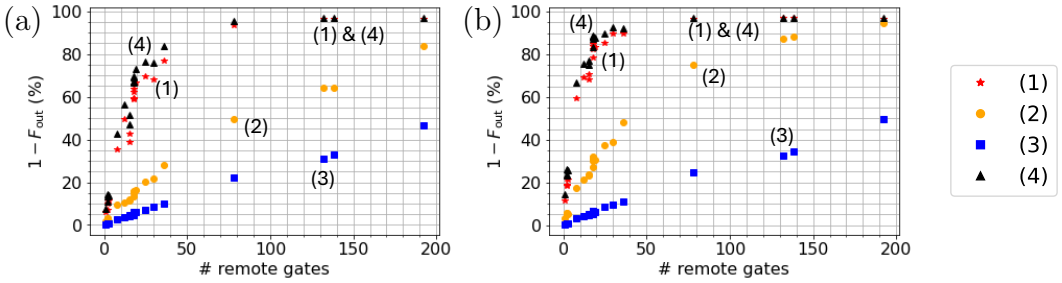


Figure 5.1: Output error, $1 - F_{\text{out}}$, as a function of the number of remote gates for 22 five-qubit MQT bench [4] quantum circuits implemented using (a) cat-comm, (b) TP-safe. The markers correspond to the output error when: (1) $\epsilon_{\text{ebit}} = 6\%$ is the only source of error; (2) $\epsilon_{\text{cnot}} = 0.4\%$ is the only source of error; (3) $r = 0.055$ Hz is the only source of error; and (4) all three types of errors are present and have the values quoted for (1)-(3). In some cases, multiple quantum circuits share the same number of remote gates and a separate data point is plotted for each. This occurs for three of the remote gate values.

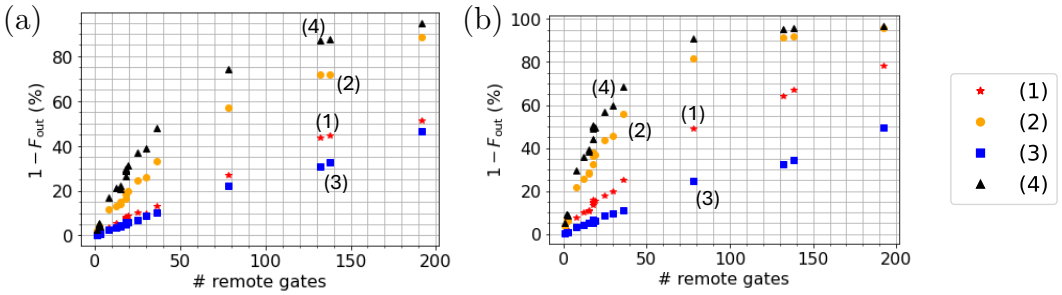


Figure 5.2: The output error as a function of the number of remote gates for 22 five-qubit MQT bench [4] quantum circuits implemented using (a) cat-comm, (b) TP-safe. The markers correspond to the output error when: (1) $\epsilon_{\text{ebit}} = 0.5\%$ is the only source of error; (2) $\epsilon_{\text{cnot}} = 0.5\%$ is the only source of error; (3) $r = 0.055$ Hz is the only source of error; and (4) all three types of errors are present and have the values quoted for (1)-(3). In some cases, multiple quantum circuits share the same number of remote gates and a separate data point is plotted for each. This occurs for three of the remote gate values.

and local gate errors are set to $\epsilon_{\text{ebit}} = 0.5\%$ and $\epsilon_{\text{cnot}} = 0.5\%$, respectively, for comparison. For memory depolarisation, $r = 0.055$Hz, as before.

Figure 5.1 shows that observation i. from Sec. 4.5.3 also holds for larger circuits. Once more, the impact of entanglement error dominates when the state-of-the-art error probability is used, followed by the impact of local gate errors, and then that of memory depolarisation. The one caveat is that, unlike for single remote gates, the impact of memory depolarisation is not negligible for some of the circuits considered. Nonetheless, memory depolarisation is still the least impactful type of error. In most cases, the difference in the impact of memory depolarisation from that of the other types of error considered remains quite large.

The magnitudes of the output error observed in Fig. 5.1 also offer some insight into what will be achievable with QDCs in the near-term. With as few as 10-20 remote gates in a circuit, the three errors considered can drive the output error to approximately 50% when cat-comm is used. An output error as high as 50% makes quantum advantage very unlikely. Even with entanglement error alone, under 20 remote gates can be tolerated before the output error exceeds 50%. For TP-safe, around five remote gates can be tolerated before the output error exceeds 50%.

When the magnitude of the entanglement error is reduced to fall within the distilled range and we impose that $\epsilon_{\text{ebit}} = \epsilon_{\text{cnot}} = 0.5\%$ (see Fig. 5.2), observation ii. from Sec. 4.5.3 is also found to be true for larger circuits. As with the single remote gates, local errors have the greatest adverse effect on the output error, followed by entanglement error and then memory depolarisation. This is most likely due to far greater number of local two-qubit gates than ebit distributions overcoming the greater probability of errors occuring in the ebits. Unlike for single remote gates, the impact of entanglement error and memory depolarisation is similar when cat-comm is used to implement all remote gates. This can be seen from the proximity of curves (1) and (3) in Fig. 5.2(a) and is explained by the fact that the larger quantum circuits that we consider here must do a variety of local operations as well as remote gates. This makes the ratio of total circuit execution time to number of ebits distributed much higher than for the single remote gates we consider in Chapter 4. As such, time-dependent memory depolarisation becomes much more consequential than it was previously, relative to

entanglement error. For TP-safe, the impacts of entanglement error and memory depolarisation on output error are less similar to each other than for cat-comm, but they are more similar to each other than is observed for single remote gates. This can be seen using curves (1) and (3) of Fig. 5.2(b), which are further apart than curves (1) and (3) in Fig. 5.2(a).

Additionally, reducing entanglement error to this extent has a significant quantitative impact on the output error. If cat-comm is used, around 30-40 remote gates could be tolerated before the output error observed in curve (4) of Fig. 5.2(a) exceeds 50%. With TP-safe, around 10-20 remote gates could be tolerated. Thus, reducing the entanglement error by an order of magnitude is likely to dramatically improve the number of quantum circuits with which potentially meaningful experiments could be done using a QDC.

Another instructive resource to consider is the number of qubits. The impact of varying the number of qubits is considered for all of the circuits in the MQT bench suite by implementing versions of the same circuits but with a different quantum register size in each case. The effect of this on the output error is shown in Figs. 5.3 and 5.4 for a selection of the MQT Bench circuits with fixed error parameters in the state-of-the-art and distilled ranges, respectively. For each set of data points, one type of error is non-zero and the other types of error are set to zero. The results for the remainder of the circuits considered can be found in Ref. [119]. As the standard error is negligible for the five-qubit circuits, which are averaged over ten runs, no such averaging is done for the other circuits, with different numbers of qubits, to avoid incurring the significant increase in runtime associated with repeating the simulation.

Figures 5.3 and 5.4 provide further evidence for the previous discussion on the relative impact of each error type. Using Fig. 5.3, we again see that, with the state-of-the-art error parameters, entanglement error has the largest adverse impact on output error, followed by local gate errors and then memory depolarisation. When entanglement error is reduced to be within the distilled range (see Fig. 5.4), we again find that the impact of local gate errors dominates. However, if all remote gates are implemented with cat-comm, we find that for many of the circuits considered, the impact of memory depolarisation can exceed that of entanglement error when the number of qubits used is high enough. Figure 5.4(a) shows this

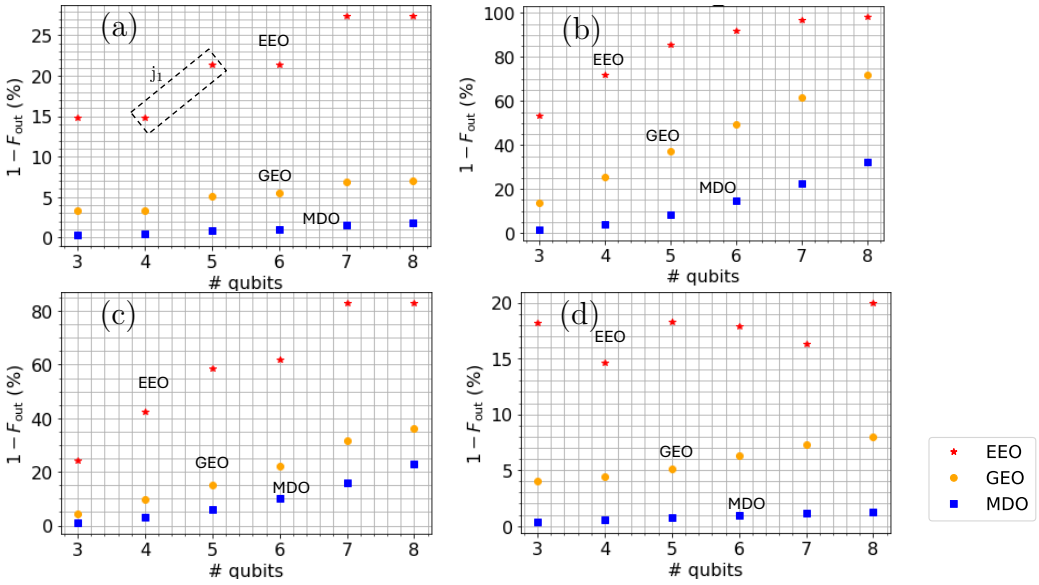


Figure 5.3: The output error, $1 - F\text{out}$, as a function of the number of qubits used to implement: (a) the Deutsch-Jozsa circuit; (b) a quantum neural network; (c) the variational quantum eigensolver (VQE) applied to portfolio optimisation; and (d) VQE with a TwoLocal ansatz applied to the max-cut problem. All circuits are taken from Ref. [4]. In plot (c) cat-comm is used to implement all remote gates and in all other cases TP-safe is used. Each set of data points shows the results when one of the error parameters (see the annotations) is non-zero, and all other errors are set to zero. The non-zero error values had the state-of-the-art values given in Table 4.1.

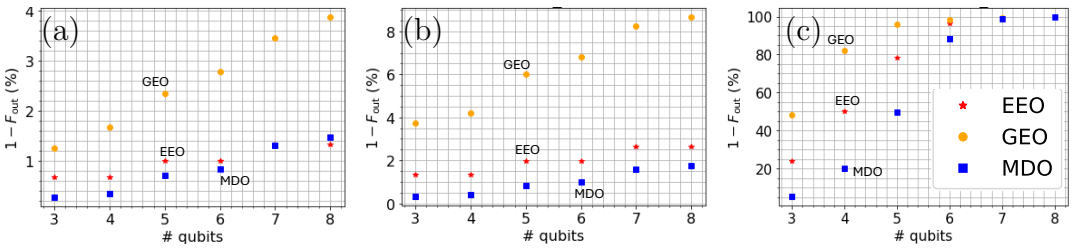


Figure 5.4: The output error, $1 - F_{\text{out}}$, as a function of the number of qubits used to implement: (a) the Deutsch-Jozsa circuit with cat-comm; (b) the Deutsch-Jozsa algorithm with TP-safe; (c) a quantum walk without ancilla qubits using TP-safe. Each set of data points shows the results when one of the error parameters (see the annotations) is non-zero, and all other errors are set to zero. The non-zero error parameters used are: $\epsilon_{\text{ebit}} = 0.5\%$ for entanglement error only (EEO); $\epsilon_{\text{cnot}} = 0.5\%$ for two-qubit gate errors only (GEO); and $r = 0.055\text{Hz}$ for memory depolarisation only (MDO). For ϵ_{ebit}, this means the distilled range is used.

happening for the Deutsch-Jozsa algorithm. Further examples can be found in Ref. [119]. This is intuitive, as increasing the number of qubits will increase the latency of the circuit, meaning qubits are more likely to undergo memory decoherence.

A less intuitive feature of Figs. 5.3 and 5.4 is the existence of discrete jumps and plateaus in the output error. In some cases, the jumps and plateaus can be easily understood. For example, consider jump j_1 in Fig. 5.3(a), which shows $1 - F_{\text{out}}$ as a function of the number of qubits for Deutsch-Jozsa circuits implemented with TP-safe. Jump j_1 occurs when increasing the number of qubits from four to five because the number of ebits increases from two to three when doing so. As such, it is natural for the entanglement error to increase and cause an increase in $1 - F_{\text{out}}$. By contrast, when we consider the five and six qubit circuits, both have the same number of ebits (three) and so we see a corresponding plateau in $1 - F_{\text{out}}$ for the curve in which only entanglement error, ϵ_{ebit}, is non-zero. Similar arguments can be used to explain the other jumps observed in Fig. 5.3(a) and Figs. 5.4(a)-(b). On the other hand, in some cases, what causes jumps and plateaus remains an open question. Consider, for example, Fig. 5.3(d), which shows $1-F_{\text{out}}$ for circuits implementing the variational quantum eigensolver (VQE) algorithm using TP-safe with increasingly many qubits. The curve for which only the entanglement error is non-zero fluctuates quite a bit, but the number of ebits remains constant at two for all data points. Therefore, unlike for the Deutsch-Jozsa algorithm, variations in $1 - F_{\text{out}}$ cannot be explained by changes in the number of ebits. What causes these fluctuations in $1 - F_{\text{out}}$ for VQE remains an open question. These features of the results allude to the fact that knowing the quantity of resources such as the number of ebits or local CNOT gates is not always enough to predict the error propagation in distributed quantum circuits. We infer that the exact structure of the compiled quantum circuit is also important to the error propagation and suggest caution against over-reliance on heuristic optimisation that considers only resource quantities when compiling distributed quantum circuits, although this remains a useful starting point. There is still much to learn about error propagation in QDCs and distributed quantum computers more generally.

5.5 Conclusion

In this chapter, we investigated a variety of distributed quantum circuits, in the presence of various types of error, using classical simulation. We found that the relative impact of entanglement errors, imperfect implementation of local gates, and time-dependent memory decoherence in larger distributed quantum circuits is qualitatively identical to what was found for single remote gates, apart from the occurrence of saturation behaviour in larger circuits as the fidelity of results reaches its minimum value. We also found that if current quantum technology could be successfully integrated to form a single QDC, a circuit containing around 10 remote gates could be implemented with cat-comm before the output error exceeded 50%. If the magnitude of the entanglement error could be reduced to the same value as local gate error, a circuit containing at least 30 remote gates could be implemented. These numbers could most likely be significantly improved by using more optimal compilation strategies.

Our results are likely to be most applicable to matter-based QPUs with photonic interconnects, although, it is possible that they could be relevant to other platforms in the future, if a practically viable method of photon to communication qubit transduction is discovered for those platforms. Viewed collectively, our results indicate that small to medium-sized experimental demonstrations of a QDC are feasible in the near to mid-term. This has been recently corroborated by an experimental implementation of a small two-QPU QDC [52], albeit with a very low entanglement distribution rate (of 9.7Hz)—to keep the entanglement error low. However, while proof-of-principle demonstrations of QDCs are within reach, our results indicate that there are unlikely to be benefits to using QDCs over monolithic devices in the near term. The magnitude of entanglement errors currently within experimental reach is simply too high for QDCs to be worthwhile.

Looking forward, it will be necessary to mitigate the impact of entanglement error to make QDCs competitive with with monolithic devices. That said, even when we improved the entanglement error by an order of magnitude, the results still indicate that it will be difficult for QDCs to be practically useful. However, the fact that entanglement error is no longer the most significant source of error in this regime is an indicator that QDCs may come into their own further in the

future. Indeed, some recent results for fault-tolerant, error-corrected distributed quantum computers [120] indicate that distributed quantum computing may be competitive with monolithic devices when enough qubits are available for entire quantum circuits to be implemented as logical operations encoded in a quantum error correction code. As a stepping stone, towards this longer-term goal, we consider the partial use of simple quantum error detection codes and entanglement distillation techniques in Chapter 6. Although it is beyond the scope of this thesis, it would also be interesting to explore the fully fault-tolerant regime in the future.

Chapter 6

Combatting noise in near-term Quantum Data Centres

6.1 Introduction

In Chapters 4 and 5, we established that entanglement error dominates other forms of error with currently obtainable error parameters. In this chapter, we propose and evaluate various near-term strategies for mitigating the impact of entanglement error in quantum data centres (QDCs). We consider two broad categories of error handling: quantum error detection (QED) and entanglement distillation, in the context of a remote CNOT gate.

We make several additions to the existing literature. Our main contributions are as follows:

- We propose the notion of localised QED encoding as a means to improve the fidelity of remote gates in the QDC setting.
- We perform a circuit-level analysis of various QED schemes.
- We compare the relative performance of QED and entanglement distillation for near-term QDCs.

The remainder of the chapter is structured as follows. In Sec. 6.2, we discuss how this chapter fits in with the existing literature. In Sec. 6.3, we introduce the schemes analysed in this paper. The error analysis conducted on these schemes

is discussed in Sec. 6.4. The results are then presented in Sec. 6.5. Finally, we conclude our discussion in Sec. 6.6.

6.2 Related Literature

For the QED portion of this chapter, in contrast to much of the previous work on QED in the computing context [121–126], we do not assume (QED) encoding is retained throughout the entire algorithm, which would require many logical gates and qubits. Instead, we propose the localised application of QED to inter-QPU, or remote, gates, in which QED encoding is performed immediately prior to a remote gate and decoding is performed during the remote gate. The most similar work to ours is Ref. [127] but they consider detecting errors only on ebits and not the arbitrary unknown state of processing qubits.

For entanglement distillation, we extend previous work [128–130] by considering the average over all possible pure, separable input states to the remote gate, rather than assuming a known state is teleported. We also explicitly consider an entire remote CNOT gate rather than only the teleportation part [128, 129] or only the distribution of ebits [130]. This impacts the results if local errors are considered.

6.3 System description

As CNOT and single qubit gates are sufficient for universal quantum computing [56], we restrict ourselves to the study of remote CNOT gates for concreteness. We first consider the basic, unencoded remote gate in Sec. 6.3.1 and then detail the additions that we make to it using QED, in Sec. 6.3.2, and entanglement distillation, in Sec. 6.3.3.

6.3.1 Unencoded case

We use 1TP, defined in Chapter 4, to implement remote gates, as shown in Fig. 6.1. For simplicity, we do not also consider the cat-comm scheme introduced in Chapter 4, however this would be an interesting task for future work.To implement a 1TP remote CNOT, the control qubit, A_{p_0}, is teleported [88, 89] to B_{c_0} on the

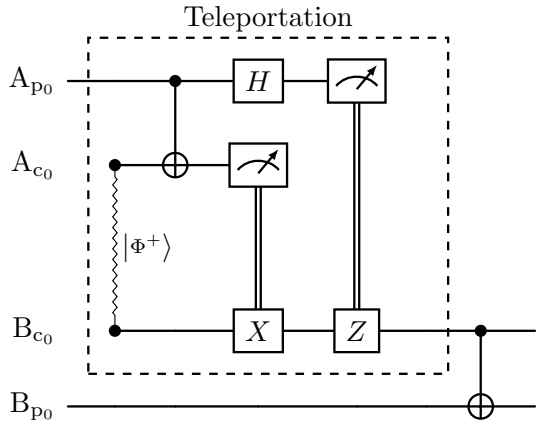


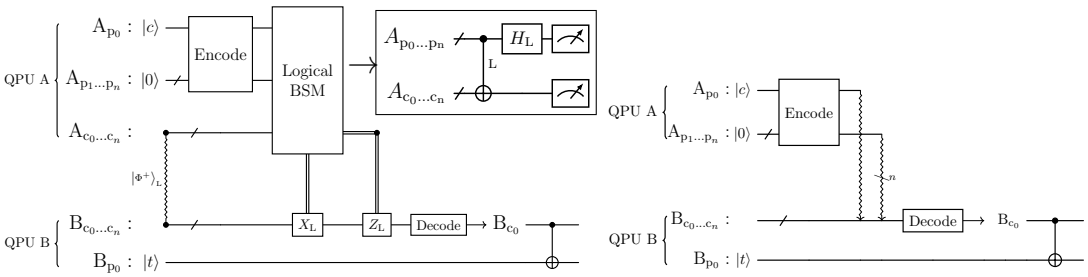
Figure 6.1: A remote CNOT gate between qubits A_{p_0} and B_{p_0} implemented using the 1TP protocol. Zigzag lines indicate the distribution of an ebit, which, ideally, in the absence of noise, has the state $|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$. Double lines indicate classical communication. The subscript c_i indicates a communication qubit with index i and p_i is a processing qubit with index i.

QPU containing the target qubit, B_{p_0}, and then the CNOT gate is conducted locally using B_{c_0} as the control qubit and B_{p_0} as the target qubit. Three different types of qubit are used: communication qubits, A_{c_0} and B_{c_0}, processing qubits, A_{p_0} and B_{p_0}, and flying qubits, which are not shown explicitly in Fig. 6.1. The flying qubits, which are typically photons, are used to distribute entanglement between the communication qubits on each QPU. The processing qubits, which may or may not be physically distinct from the communication qubits, are used much like qubits in a monolithic setting and do not interact directly with flying qubits.

6.3.2 QED schemes

We consider two broad ways of integrating QED into a 1TP remote CNOT gate, as shown in Fig. 6.2. The first scheme, shown in Fig. 6.2(a), is referred to as fully-coded 1TP (FC-1TP) and works as follows:

1. We first distribute a logical ebit, of the form $|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle_L + |11\rangle_L)$, between both QPUs involved in the remote gate, where $|0\rangle_L$ and $|1\rangle_L$ are the logical computational basis eigenstates encoded in the QED code of choice.



(a) (b)

Figure 6.2: 1TP remote CNOT gate using (a) fully-coded 1TP (FC-1TP) and (b) partially-coded 1TP (PC-1TP). The subscript L refers to a logical version of the operation or state. We omit the L for the measurements in (a) but both measurements are logical measurements. Double lines refer to classical communication, squiggly lines to the distribution of an ebit and squiggly arrows to quantum teleportations from the tail of the arrow to its head. $|c\rangle$ and $|t\rangle$ are arbitrary quantum states, which we assume to be separable from each other and pure in this work. If an error is detected during the decoding step then the final CNOT is not done and the entire process must be restarted.

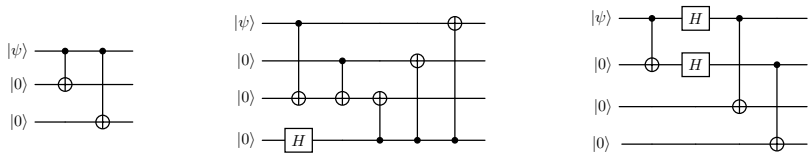
This is done, using the method discussed in Ref. [65] for encoded quantum repeaters, as follows:

(a) Qubits $A_{c_0 \ldots c_n}$ are initially encoded in the logical state $|+\rangle_{\mathrm{L}} = \frac{1}{\sqrt{2}}(|0\rangle_{\mathrm{L}} + |1\rangle_{\mathrm{L}})$ and qubits $B_{c_0 \ldots c_n}$ are encoded in the logical state $|0\rangle_{\mathrm{L}}$.

(b) A logical remote CNOT gate is done between qubits $A_{c_0 \ldots c_n}$ and $B_{c_0 \ldots c_n}$. In our case, we only consider the repetition code for the first scheme and so $|0\rangle_{\mathrm{L}} = |000\rangle$ and $|1\rangle_{\mathrm{L}} = |111\rangle$ and the logical remote CNOT gate is done transversally using additional communication qubits, not shown in Fig. 6.2(a), to implement each of the physical remote gates. However, it is easy to generalise the same concepts to different QED codes.

2. Then, using $A_{p_1 \ldots p_n}$, we apply local gates to encode qubit $\mathrm{A}_{\mathrm{p}_0}$.

3. After that, a logical BSM is applied, as shown in the box in the top-right corner of Fig. 6.2(a). Firstly, a logical CNOT is conducted between $A_{p_0 \ldots p_n}$ and $A_{c_0 \ldots c_n}$. Then, a logical X-basis measurement is done on $A_{p_0 \ldots p_n}$ and a logical Z-basis measurement is done on $A_{c_0 \ldots c_n}$.



(a) (b) (c)

Figure 6.3: Encoding circuits for (a) the three-qubit repetition code, (b) the Leung-Nielsen-Chuang-Yamamoto (LNCY) code [5] code using the encoder from [6], (c) a simpler encoding of the LNCY. $|\psi\rangle$ is an arbitrary quantum state.

4. The results of the measurements are sent from QPU A to QPU B and the local X and Z operations needed to complete the teleportation protocol are done on $B_{c_0 \ldots c_n}$.

5. Finally, decoding is done and a physical CNOT gate is done between B_{c_0} and B_{p_0}. If errors are detected during decoding or the logical BSM, the CNOT gate is not done and the entire process is repeated from the beginning.

In the second scheme, shown in Fig. 6.2(b), logical ebits are not used. Instead, $\mathrm{A}_{\mathrm{p}_0}$ is encoded into a QED code, using $A_{p_1 \ldots p_n}$, as before, and then qubits $A_{p_0 \ldots p_n}$ are individually teleported to $B_{c_0 \ldots c_n}$, using additional ebits not shown in Fig. 6.2. After that, decoding is done, as in FC-1TP, and a local, unencoded, CNOT gate is done locally between B_{c_0} and B_{p_0}. We refer to this scheme as partially-coded 1TP (PC-1TP).

PC-1TP has the advantage of avoiding the need for any logical operations. This allows it to be used with a larger class of codes, such as the [[4, 1, 2]] Leung-Nielsen-Chuang-Yamamoto (LNCY) code [5], for which not all of the logical gates needed for FC-1TP can be implemented transversally. For example, one can verify with manual calculation that, for the LNCY code, no transversal single-qubit logical Hadamard gate exists. PC-1TP is also trivial to extend to any other QED schemes without the need to worry about the fact that different QED codes use different logical gate implementations.

Figure 6.3 shows the encoding circuits for the specific codes used in this paper. Figure 6.3(a) encodes an arbitrary quantum state into the three-qubit repetition code, which we shall refer to as 3QRC. This has the logical encoding $|0\rangle_{\mathrm{L}} = |000\rangle$

and $|1\rangle_{\mathrm{L}} = |111\rangle$ on the basis states. An arbitrary state, $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, where $\alpha, \beta \in [0, 1]$ are normalised such that $|\alpha|^2 + |\beta|^2 = 1$, is encoded as $|\psi\rangle_{\mathrm{L}} = \alpha|000\rangle + \beta|111\rangle$. We consider both FC-1TP and PC-1TP with 3QRC and refer to the former as FCRC and the latter as PCRC. Figs. 6.3(b) and 6.3(c) encode the [[4, 1, 2]] LNCY [5], which, up to qubit relabelling, has the logical eigenstates $|0\rangle_{\mathrm{L}} = \frac{1}{\sqrt{2}}(|0000\rangle + |1111\rangle)$ and $|1\rangle_{\mathrm{L}} = \frac{1}{\sqrt{2}}(|0101\rangle + |1010\rangle))$. These are the $\{|00\rangle_{\mathrm{L}}, |10\rangle_{\mathrm{L}}\}$ logical basis states of the four-qubit code [123]. The encoding circuit shown in Fig. 6.3(c) is slightly more resource efficient, requiring fewer gates, but we include the encoding shown in Fig. 6.3(b) too, for the sake of comparison. We refer to the schemes shown in Figs. 6.3(b) and 6.3(c) as 4QED, for four-qubit QED, and SS, for shortened Shor, respectively. Both of these names could apply to either scheme, as the two schemes are equivalent up to relabelling and so both schemes are four-qubit QED codes which can be thought of as shortened versions of the Shor code [30] but, to distinguish them, we uniquely apply the labels 4QED and SS to Figs. 6.3(b) and 6.3(c), respectively. As previously discussed, it is challenging to implement FC-1TP for the LNCY code, and so we consider only PC-1TP for both 4QED and SS.

Unitary decoding of an arbitrary encoded state is done by running the encoding circuits in reverse. If the encoding gates all commute, as they do for 3QRC, then the encoding circuit can be used unchanged. For the 3QRC decoder, we integrate error detection into the decoder similarly to the proposal in Ref. [131]. For 4QED and SS, we instead detect errors prior to decoding using a stabiliser measurement [6] and then run the unitary decoding circuits. This requires the use of additional, ancilla qubits. The decoding/error detecting circuits used are shown in Fig. 6.4. The X stabiliser measurements in Figs. 6.4(b) and 6.4(c) detect Z errors and the Z stabiliser measurements detect X errors [6, 56].

6.3.3 Entanglement distillation schemes

To implement a remote CNOT gate with entanglement distillation, we use the unencoded scheme shown in Fig. 6.1 but apply entanglement distillation to the ebits. Two entanglement distillation schemes are considered, the scheme proposed in Ref. [7], termed BBPSSW, and that of Ref. [8], termed DEJMPS, hereafter.

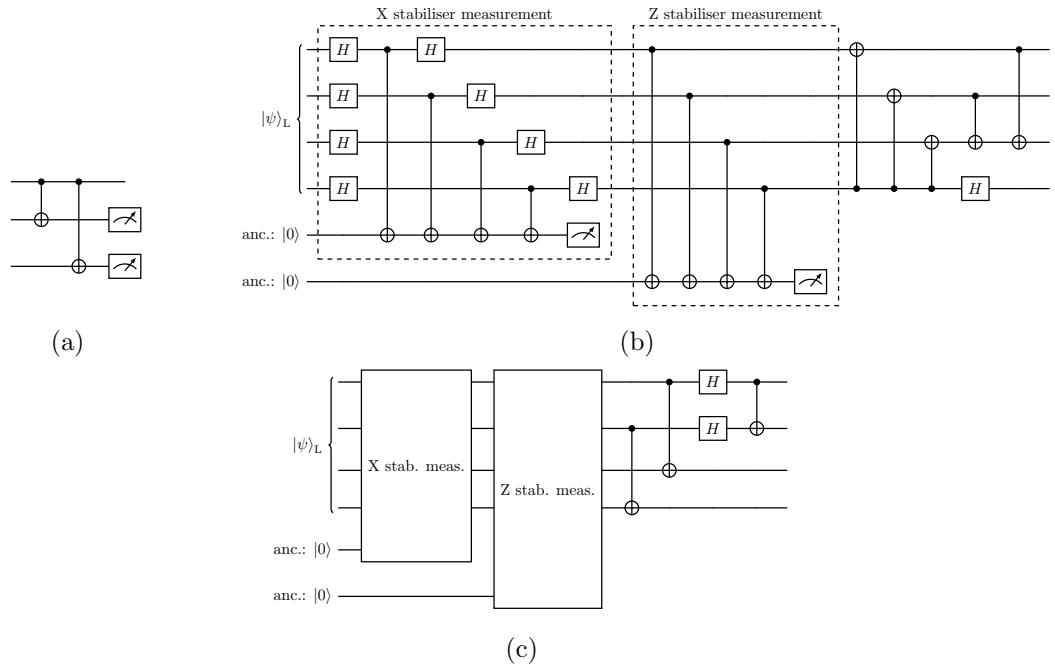


Figure 6.4: Decoding/error detection circuits for (a) 3QRC, (b) 4QED, and (c) SS. The X and Z stabiliser measurements indicated in (c) are identical to those explicitly shown in (b). Qubits labelled 'anc.' are ancilla qubits. Everything after the Z stabiliser measurement in (b) and (c) is only done if no error was detected during the X and Z stabiliser measurements. If an error is detected, the result is discarded and the entire operation must be discarded.

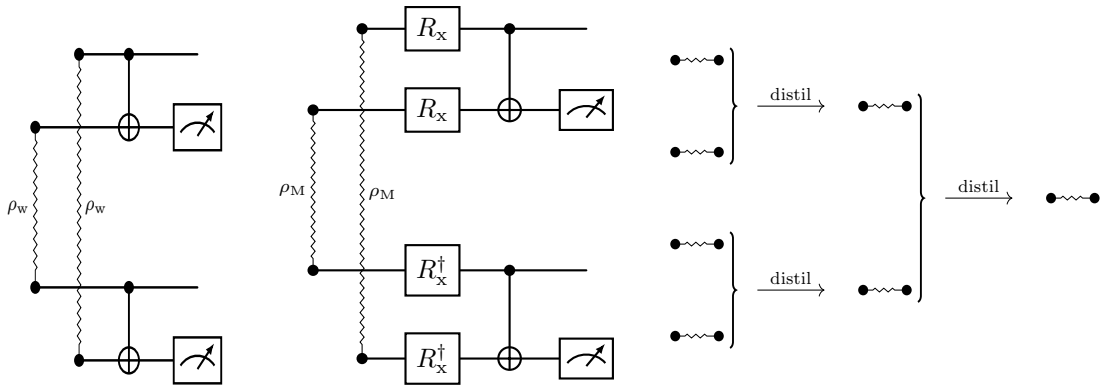


Figure 6.5: The (a) BBPSSW, (b) DEJMPS and (c) general two-round entanglement distillation schemes. BBPSSW assumes a Werner state input, which can be enforced with local operations, although this is not done here, while DEJMPS allows a more general input. For both schemes, the circuit is run repeatedly until both of the measurements shown give the same result. The squiggly lines indicate the distribution of the state $|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$, in the ideal case where no noise is present. In reality, noise will be present and for BBPSSW, it is assumed that this noise has the Werner form, as discussed in the main text. For DEJMPS, no such assumptions are made and the state ρ_{M} can instead be distributed, which is an arbitrary mixture of the different Bell states $|\Phi^{\pm}\rangle = \frac{1}{\sqrt{2}}(|00\rangle \pm |11\rangle)$, $\left|\Psi^{\pm} = \frac{1}{\sqrt{2}}(|01\rangle \pm |10\rangle)\right\rangle$. R_x is a $\frac{\pi}{2}$ rotation about the x axis and $R_x^{\dagger}$ is the $-\frac{\pi}{2}$ rotation. (a) and (b) show just one round of entanglement distillation here but the fidelity can be further improved by further distilling the outcome as illustrated in (c).

The schemes are shown in Figs. 6.5(a) and 6.5(b), respectively. The schemes differ primarily in the assumptions that they make about the initial entangled state. BBPSSW is conducted by first distributing two ebits, which are assumed to be in the Werner state, defined as follows:

$$\rho_{\mathrm{w}} = F_{\mathrm{w}} \left|\Phi^+\right\rangle\left\langle\Phi^+\right| + \frac{1 - F_{\mathrm{w}}}{3}\left(\left|\Phi^-\right\rangle\left\langle\Phi^-\right| + \left|\Psi^+\right\rangle\left\langle\Psi^+\right| + \left|\Psi^-\right\rangle\left\langle\Psi^-\right|\right), \quad (6.1)$$

where $F_{\mathrm{w}} \in [0, 1]$, $|\Phi^{\pm}\rangle = \frac{1}{\sqrt{2}}(|00\rangle \pm |11\rangle)$, and $|\Psi^{\pm}\rangle = \frac{1}{\sqrt{2}}(|01\rangle \pm |10\rangle)$. This assumption can be enforced using random single-qubit gates [7], but, in this work, we assume that the state of the initial ebit already has the above form, as we consistently use the Werner state to model entanglement noise throughout this work. BBPSSW also requires $F_{\mathrm{w}} \geq 0.5$.

The second scheme, DEJMPS, instead assumes that the ebits have the more

general state

$$\rho_{\mathrm{M}} = A\left|\Phi^{+}\right\rangle\left\langle\Phi^{+}\right| + B\left|\Psi^{-}\right\rangle\left\langle\Psi^{-}\right| + C\left|\Psi^{+}\right\rangle\left\langle\Psi^{+}\right| + D\left|\Phi^{-}\right\rangle\left\langle\Phi^{-}\right|, \tag{6.2}$$

where A, B, C, $D \in [0,1]$, $A+B+C+D=1$, and $A \geq 0.5$. However, for consistency with our error analysis of the QED schemes, we again assume that the initial, pre-distillation ebits have the Werner state in Eq. (6.1).

After the entanglement distribution step, DEJMPS applies the operation $R_{\mathrm{X}} \otimes R_{\mathrm{X}}^{\dagger}$, to each ebit, where R_{X} is a $+\frac{\pi}{2}$ rotation about the x axis. This swaps B and D in Eq. (6.2). After that, the same local operations and measurements as BBPSSW are used.

Both schemes start with two pairs of entangled states entangled between two QPUs, apply CNOT gates locally on their share of the entangled states, and then measure the target qubits. The measurement results are compared and if they do not have the same value then the unmeasured qubits are discarded. This process distinguishes between the $|\Phi^{\pm}\rangle$ and $|\Psi^{\pm}\rangle$ states [7]. The entire process is repeated until both measurement results are the same.

It is possible to improve the fidelity by nesting the entanglement distillation process so that already distilled ebits are distilled again. This is shown in Fig. 6.5(c). DEJMPS is known to perform better than BBPSSW when multiple nested rounds of entanglement distillation are done [8] and so we consider nesting only for DEJMPS. In the nested DEJMPS protocol that we consider, four ebits are initially distributed sequentially, waiting for one distribution to finish before starting the next. This is not optimal. Clearly, parallelisation could be done here, but, for ease of simulation, we have not done so. After being distributed, the four initial ebits are distilled into two higher-fidelity ebits. Those two distilled ebits are then distilled again to make a single ebit. If either distillation fails, we repeat the entire process again, creating four ebits and distilling them until the entire protocol succeeds. Again this may not be optimal. If one of the distillations is successful and the other fails, we could retain the successfully distilled ebit and discard only the unsuccessfully distilled one. This would arguably be more efficient but risks memory depolarisation damaging the previously distilled ebit while it waits for another ebit to be successfully distilled. To avoid such complications, we discard both pairs. We refer to the four ebit nested DEJMPS protocol as DEJMPS4 to

distinguish it from the single-round case where we start with two ebits and distill them into a single ebit, which we call DEJMPS2.

6.4 Error analysis

We subject the various implementations of a remote CNOT gate discussed in Sec. 6.3 to various types of error. We do this using discrete event simulation, which is discussed in Sec. 6.4.1. The noise models used are discussed in Sec. 6.4.2. We also calculate the success rates of the different implementations, as discussed in Sec. 6.4.3.

6.4.1 Discrete event simulation

The majority of the results in this work are taken using discrete event simulation with NetSquid [66] and the related dqc_simulator package we developed [76]. This means that time, t, does not progress continuously but is instead a discrete variable, which increases in increments only when something, an event, happens. For example, a classical message arriving at the receiver would be an event and only when the message arrives would we apply the memory decoherence accrued when waiting for that message. All classical and quantum operations we consider have an associated duration, and when the operation finishes, we increment t by that amount. Some operations also have associated noise models, which are implemented using quantum channels. Further details on the channels used are provided in Sec. 6.4.2. The quantum state of the system is tracked by grouping together all qubits that are entangled with each other and tracking the density matrix for each group of qubits after a given event, re-assigning qubits to new groups to represent changes in entanglement, where appropriate.

Some abstractions and constraints are used for convenience of simulation. Firstly, ebits are generated at a black box central source in the Werner state, given by Eq. (6.1) and the constituent qubits travel to the QPUs. Upon arrival, the qubits are treated as communication qubits and we do not explicitly model the transduction between communication qubits and photons. The noise generated by the various subprocesses that would occur in real experiments is amalgamated

into Eq. (6.1). Similarly, the latency of the entire process is amalgamated into a single parameter, t_{dist}, representing the entanglement distribution time. Secondly, the remote gates or individual teleportations used by the FC-1TP and PC-1TP, respectively are done sequentially, due to limitations on the simple scheduler used. For both entanglement distillation schemes, ebit distribution is again done sequentially. With more advanced scheduling, parallelisation would be possible for ebit distribution, remote gates and teleportations. Finally, we neglect the classical processing time needed for classical logic. For example, the time taken to interpret classical messages or recognise that ebits have arrived at a QPU is neglected.

6.4.2 Noise models

In addition to the initial ebit noise modelled by Eq. (6.1), we model other sources of noise in our system using several variants of the depolarising noise channel [56]. First, consider single-qubit gates. For a multi-qubit system in state ρ_{in}, acting the unitary operation U_i on qubit i results in the following transformation:

$$\rho_{\text{in}} \to (1-\epsilon_{\text{sg}})U_i\rho_{\text{in}}U_i^\dagger + \frac{\epsilon_{\text{sg}}}{2}\text{Tr}_i(\rho_{\text{in}}) \otimes \mathbb{1}_i, \tag{6.3}$$

where ϵ_{sg} is the probability of error occurring for single-qubit gates, Tr_i is the partial trace with respect to qubit i, and $\mathbb{1}_i$ is its corresponding identity matrix.

For a two-qubit gate, $U_{i,j}$, acting on qubits i and j, the channel has the following form [105]:

$$\rho_{\text{in}} \to (1-\epsilon_{\text{tg}})U_{i,j}\rho_{\text{in}}U_{i,j}^\dagger + \frac{\epsilon_{\text{tg}}}{4}\text{Tr}_{i,j}(\rho_{\text{in}}) \otimes \mathbb{1}_{i,j}, \tag{6.4}$$

where ϵ_{tg} is the probability of error for two-qubit gates, $\text{Tr}_{i,j}$ is the partial trace over qubits i and j, and $\mathbb{1}_{i,j}$ is the identity matrix on i and j.

We also consider memory depolarisation, which is acted on qubits individually as time passes. This is done using the channel

$$\rho_{\text{in}} \to e^{-\Delta t\, r}\rho_{\text{in}} + \frac{(1-e^{-\Delta t\, r})}{3}(X_i\rho_{\text{in}}X_i + Y_i\rho_{\text{in}}Y_i + Z_i\rho_{\text{in}}Z_i), \tag{6.5}$$

where i is the qubit being acted on; Δt is the time since the last event; X_i, Y_i, and Z_i are the Pauli operations acting on qubit i; and r is the memory depolarisation rate, which is a parameter of the hardware.

As detailed in Sec. 6.4.1, we use event-based simulation, and so any physical process, such as a gate or the sending of a classical message has an associated duration. After each event, we apply Eq. (6.5) to all relevant qubits.

Finally, we impose errors during Z-basis measurements on qubit i using the bit-flip channel:

$$\rho_{\text{in}} \to (1-\epsilon_{\text{m}})\rho_{\text{in}} + \epsilon_{\text{m}} X_i \rho_{\text{in}} X_i, \tag{6.6}$$

where ϵ_{m} is the probability of a measurement error. We construct any X-basis measurements using a Hadamard followed by a Z-basis measurement, so all of our measurements are Z-basis measurements.

6.4.3 Success probability

As well as the noise modelled in the previous section, there is also an associated latency cost to QED and entanglement distillation. To indirectly evaluate this, we calculate the success probability, p_{s}, of a given attempt of the DEJMPS or 4QED/SS protocols as a function of F_{w}. For simplicity, we ignore other sources of error in this section.

For DEJMPS, the following expression is provided in Ref. [8]:

$$p_{\text{s}} = (A+B)^2 + (C+D)^2. \tag{6.7}$$

For multiple rounds of DEJMPS, we must recursively update the values of A, B, C and D in Eq. (6.7). We do this using the the following expressions from Ref. [8]:

$$\begin{aligned} \tilde{A} &= \frac{A^2+B^2}{p_{\text{s}}}, \\ \tilde{B} &= \frac{2CD}{p_{\text{s}}}, \\ \tilde{C} &= \frac{C^2+D^2}{p_{\text{s}}}, \\ \tilde{D} &= \frac{2AB}{p_{\text{s}}}, \end{aligned} \tag{6.8}$$

where $\tilde{A}$ to $\tilde{D}$ are the average post-distillation values of A to D.

For 4QED/SS, we find

$$p_{\text{s}} = F_{\text{w}}^4 + \frac{(1-F_{\text{w}})^4}{27} + 2F_{\text{w}}^2(1-F_{\text{w}})^2, \tag{6.9}$$

through exhaustive consideration of the all scenarios in which no error occurs or errors that occur remain undetected; see Appendix C. The latter stipulation is important and applies to DEJMPS as well. Errors can occur in a "successful" entanglement distillation or error detection process. For 4QED/SS, undetected errors are caused by even numbers of the same type of error, as in such cases, the parity with respect to the relevant error type is unchanged and so the stabiliser measurement results are indistinguishable from when no error had occurred.

6.5 Numerical results

We evaluate the performance of the schemes introduced in Sec. 6.3 using a classical, event-based simulator that we built in Python 3.9 using the NetSquid library [66] and the related dqc_simulator package that we wrote [76], see Sec. 6.4.1. All simulations were run on a high-end desktop computer with a 2.5 GHz 11th generation intel core i7-11700 processor and 32 GB of RAM. The primary figure of merit considered is the fidelity, F_{out}, of the output state from the remote gate, relative to the ideal output state in the absence of noise. Where possible, we use the hardware parameters from Quantinuum's H1 trapped-ion device [132] because at the time the results were taken, this gave the best values that we could find for the parameters that we consider. Quantinuum's newer H2 device has similar parameters, but with more qubits, which was not relevant to our analysis. When values could not be readily found for the H1 device, numbers from IonQ Aria [110], a competing commercial trapped-ion device, are used. Numbers related to the link between QPUs are sampled from those reported by Stephenson et al. at the University of Oxford [109]. Overall, the parameters used are intended to approximate what could be achieved with with current state-of-the-art technology, if that technology could be successfully integrated into a single device.

The values we use for the durations of different operations, and their corresponding references, are listed in Table 6.1. The single-qubit gate time and two-qubit gate time used are sampled from the specifications for IonQ Aria [110], a commercial trapped-ion device. The measurement time is estimated from the two-qubit gate time, based on the fact that Refs. [21, 111] assert that the measurement time is five to ten times larger than the two-qubit gate time for trapped-ion

Table 6.1: The durations of all operations considered.

Parameter	Value	Source
Single-qubit gate time	135 μs	[110]
Two-qubit gate time	600 μs	[110]
Measurement time	6 ms	Inferred from [21, 110, 111]
Classical comm. time	10^{-8} s	[109, 133]
Ebit. distr. time	$\frac{1}{182}$ s	[109]

quantum computers. We therefore choose the measurement time to be ten times the two-qubit gate time used. The classical communication time is taken to be the time to cross an optical fibre at $2 \times 10^8\,\mathrm{ms}^{-1}$ [133]. The value used for classical communication time neglects any classical processing times but gives a rough estimate of the classical communication times. For the ebit distribution time, t_{dist}, we use $t_{\mathrm{dist}} = \frac{1}{R_{\mathrm{dist}}}$, where $R_{\mathrm{dist}} = 182$ Hz is the average entanglement distillation rate found in Ref. [109].

Based on our discrete-event simulations, we find that the ability of the schemes introduced in Sec. 6.3 to mitigate entanglement error depends on their input states. It is important then to account for the role played by the input state. The input state to a given remote gate within a distributed circuit will not generally be known, so we consider the average over all pure, separable input states. The assumption of pure, separable input states limits the applicability of our findings to remote gates for which the control and target qubits are not initially entangled with any other qubits or each other, but greatly simplifies the problem. Averaging is done by the Monte Carlo method with uniform sampling of the input states of the control and target qubits from the Bloch sphere. We denote the average output fidelity over such input states as $\overline{F}_{\mathrm{out}}$. $40,000$ data points are taken for each average value computed, with the exception of the trivial case, where no error is present, and the unencoded case in the absence of local errors. In the trivial case, we take only 40 values, which is enough to verify that the output fidelity is perfect, meaning $F_{\mathrm{out}} = 1$, as expected. In the unencoded case with no local errors, we use the analytical values given by Eq. (4.5), which is input state independent.

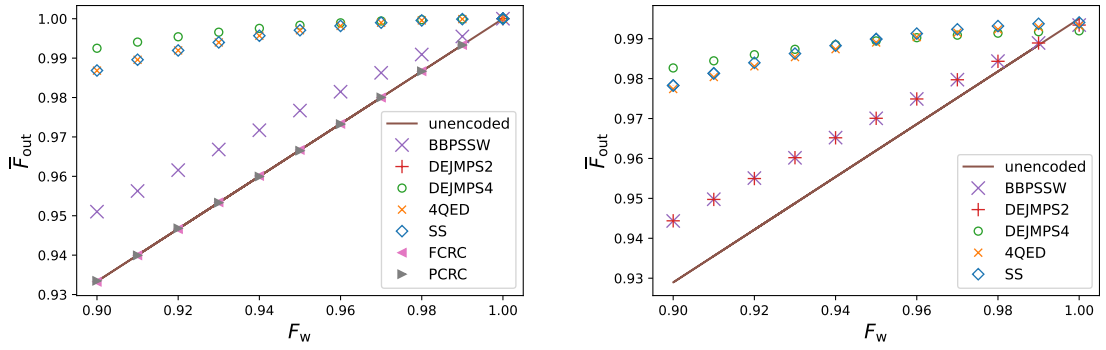


Figure 6.6: $\overline{F}_{\text{out}}$, the output fidelity averaged over all possible pure, separable input states to a remote CNOT gate, as a function of F_{w} for (a) $\epsilon_{\text{sg}} = \epsilon_{\text{tg}} = \epsilon_{\text{m}} = r = 0$, (b) $\epsilon_{\text{sg}} = 1.8 \times 10^{-5}$, $\epsilon_{\text{tg}} = 9.7 \times 10^{-4}$, $\epsilon_{\text{m}} = 2.33 \times 10^{-3}$ and $r = 0.055$Hz. With the exception of the data point at $F_{\text{w}} = 1$ in (a), which is averaged over 40 values, and the unencoded curve in (a), which is calculated using Eq. (4.5), each data point is averaged over 40,000 F_{out} values generated with different input states uniformly sampled from the Bloch sphere of the control and target qubits. The exception is made for $F_{\text{w}} = 1$ in (a) because there are no errors and so all input states should yield the ideal fidelity of $F_{\text{out}} = 1$. Therefore, to save computational resources, we consider only enough values to be verify that this is the case and include the data point only for completeness. The exception is made for the unencoded curve in (a) because the behaviour is exactly described by Eq. (4.5). The different curves represent the results for the different error mitigation schemes considered in Secs. 6.3.2 and 6.3.3. As discussed in the main text, BBPSSW and DEJMPS are the entanglement distribution schemes from Refs. [7] and [8], respectively; DEJMPS2 is two-ebit DEJMPS; DEJMPS4 is four-ebit DEJMPS; 4QED and SS are the LNCY [5] code implemented using Fig. 6.3(b) and Fig. 6.3(c), respectively. The standard error is too small to be visible.

To obtain an upper bound on performance, we first consider how each scheme performs in the presence of entanglement errors only. We consider the parameters $F_{\text{w}} \in [0.90, 0.99]$, $\epsilon_{\text{sg}} = \epsilon_{\text{tg}} = \epsilon_{\text{m}} = r = 0$, and plot $\overline{F}_{\text{out}}$ as a function of F_{w}. The results of this analysis are shown in Fig. 6.6(a). From Fig. 6.6(a), the following observations can be made:

1. The repetition code schemes, FCRC and PCRC, are indistinguishable from each other in the presence of entanglement error. It is interesting that there is no intrinsic entanglement error reduction benefit to using the more complicated FCRC scheme over the simpler PCRC scheme in our scenario. The PC-1TP method, shown in Fig. 6.2, benefits from a reduced gate count,

greater simplicity and easier generalisation to other QED codes. For this reason, we use this scheme for the remaining codes investigated.

2. Even in the absence of local errors, neither FCRC or PCRC shows any benefit, on average, over the unencoded remote gate. From inspection of the data, significant advantage is found over the unencoded case for certain input states but for others they actually do worse than the unencoded case. This is mainly due to the fact that the repetition code used detects bit flip errors as opposed to phase flip ones. More details on this are provided in Appendix D. However, we have been unable to infer a clear pattern as to which input states do better than others.

3. Entanglement distillation and both four-qubit error detection schemes, 4QED and SS, all provide a clear advantage over the unencoded case. The greatest advantage is provided by DEJMPS4 followed by 4QED and SS, with the least advantage offered by BBPSSW. It can be verified analytically that DEJMPS2 yields an identical entanglement fidelity to BBPSSW in this scenario and so is not shown.

4. 4QED and SS give essentially identical results. The difference between any two results always falls within the standard error range of the results. This is to be expected, as in the absence of local errors, there is very little to distinguish the encoders and decoders of the two schemes.

We also consider the more realistic setting where finite local errors are present in the circuit. We use the parameters $F_{\mathrm{w}} \in [0.90, 0.99], \epsilon_{\mathrm{sg}} = 1.8 \times 10^{-5}$, $\epsilon_{\mathrm{tg}} = 9.7 \times 10^{-4}$, $\epsilon_{\mathrm{m}} = 2.33 \times 10^{-3}$ and $r = 0.055\,\mathrm{Hz}$, where the fixed parameter values for ϵ_{tg} and r are based on those available in real trapped-ion hardware [110, 132]. For everything except r, the values for Quantinuum's H1 processor [132] are used. For r, the midpoint of the quoted range for $\frac{1}{T_1}$ is used with T_1 taken from the data for IonQ Aria [110]. The parameters from the H1 and IonQ Aria machines are used because they represent the state of the art for these parameters on commercial hardware at the time of writing. Again, $\overline{F}_{\mathrm{out}}$ is plotted for F_{w} in the previously stated range to produce Fig. 6.6(b). Based on Fig. 6.6(b), we observe:

1. As was the case previously, both of the four-qubit QED schemes considered, 4QED and SS, outperform both two-ebit entanglement distillation schemes. DEJMPS4, however, does a little better than 4QED and SS for $F_{\mathrm{w}} < 0.95$ and very slightly worse thereafter. The discrepancy between DEJMPS4 and 4QED/SS is always small for all considered F_{w} values.

2. 4QED, SS and DEJMPS4 have a very clear advantage over the unencoded case until around $F_{\mathrm{w}} \approx 1$, at which point they perform similarly to the unencoded case. Therefore, there is a clear benefit to their use with current or near-term hardware.

3. DEJMPS2 and BBPSSW are almost identical, with at most 0.01% between them and neither scheme being consistently better for all F_{w}.

4. BBPSSW and DEJMPS2 outperform the unencoded case for $F_{\mathrm{w}} \lessapprox 0.99$ but offer a much smaller advantage than the DEJMPS4, 4QED and SS.

5. SS slightly outperforms 4QED in the presence of the local error values considered, but the discrepancy between the two schemes remains very low. Overall, it seems that the slightly less gate efficient encoder and decoder used in 4QED makes very little difference in this scenario because the magnitude of the local errors are too low for a few gates to have a significant impact.

From the $\overline{F}_{\mathrm{out}}$ results alone there is little to distinguish between 4QED/SS, which are similar enough to be discussed together here, and DEJMPS4. However, there are other practical considerations to consider. The first of these is that 4QED/SS requires more qubits to implement than DEJMPS4, as both schemes need four communication qubits available but 4QED/SS requires an additional three processing qubits on one QPU, with which to encode the control qubit prior to teleportation.

Another area in which there is some distinction is the impact of each scheme on the latency. A key facet of this is the fact that failed rounds of DEJMPS only require communication qubits to be discarded and so do not destroy any quantum information required to complete the algorithm, which is stored in the processing qubits. The ebits used in DEJMPS are a consumable resource and the only cost to

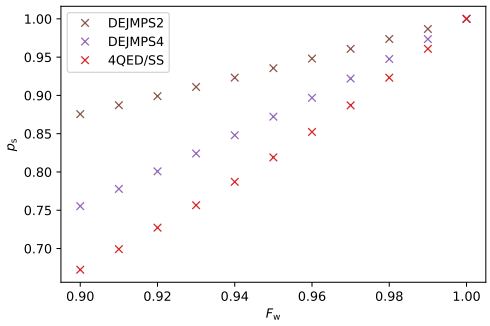


Figure 6.7: Success probability for different error mitigation schemes with varying F_w and $\epsilon_\mathrm{sg} = \epsilon_\mathrm{tg} = \epsilon_\mathrm{m} = r = 0$.

discarding them is time. By contrast, if 4QED/SS fails then the entire algorithm must be restarted as there is a detected error in the processing qubits. Therefore, if the success probability, p_s, of 4QED/SS and DEJMPS is comparable then the latency cost of 4QED/SS will be much higher for larger circuits.

Using Eqs. (6.7) and (6.9), we find that DEJMPS actually has a higher p_s than 4QED/SS, with DEJMPS2 being up to 30.2% greater than 4QED and DEJMPS4 being up to 12.4% greater. We plot p_s as a function of F_w, with $\epsilon_\mathrm{sg} = \epsilon_\mathrm{tg} = \epsilon_\mathrm{m} = r = 0$, in Fig. 6.7. From Fig. 6.7, we can see that, for all F_w values, p_s is highest for DEJMPS2, followed by DEJMPS4 with 4QED/SS having the lowest rate. As such, in the particular case considered here, entanglement distillation introduces a lower latency than localised QED encoding. That said, QED schemes provide a pathway solution between the current state of the art, with no encoding, and the eventual fault-tolerant schemes. As such, their analysis and implementation could offer hints on how to move forward.

6.6 Conclusion

In this work, we applied quantum error detection to remote gates as a means of improving fidelity and investigated the performance of those schemes and entanglement distillation in increasing the fidelity of remote gates within the quantum data centre context. Taken together, our results suggest that, of the error mitigation schemes considered, entanglement distillation is most suitable for

use in resource-constrained, near-term applications, as entanglement distillation is able to improve the average output fidelity better or comparably to the error detection schemes investigated, with fewer qubits and a lower latency cost. For the former, an alternative, not considered in this work, would be to perform error detection only on the ebit resources and decode prior to beginning teleportation, which may offer better performance. That solution is conceptually very similar to entanglement distillation.

There is a trade-off between the better fidelity improvement of four-ebit, two-round DEJMPS entanglement distillation and the faster implementation and lower qubit cost of two-ebit, one-round DEJMPS. However, the significant improvement in output fidelity of four-ebit DEJMPS relative to two-ebit DEJMPS most likely makes it worth doing at least two rounds of entanglement distillation.

The [[4, 1, 2]] LNCY [5] also shows a clear improvement over the unencoded case and future investigation of alternative error detection codes or optimisations is merited. However, the three-qubit repetition code is unlikely to be useful for quantum computing purposes in the quantum data centre context.

Chapter 7

Conclusion and Future Work

In this thesis, we explored noise in the quantum data centre paradigm of distributed quantum computing.

We started off by investigating the impact of noise on isolated remote CNOT gates, using the output fidelity as the primary figure of merit. Several ways of implementing remote gates were considered, which we dubbed cat-comm, 1TP, 2TP and TP-safe. We found that simple first-order approximations to error analysis failed to accurately capture the behaviour of even this simple system with quantitative differences in the output fidelity as high as 60% between the simulated and approximate results in some cases.

We improved upon the basic first-order approximations with exact analytical solutions for the output fidelity of 1TP and cat-comm, in the constrained scenario where only entanglement error is present and the processing qubits involved in the remote gate have separable states prior to the remote gate. Excellent agreement was observed between these analytical results and the output for the event-based simulator, dqc_simulator, that we developed and used to take the majority of the results in this thesis.

With the aid of dqc_simulator, we extended our analysis to include all of the remote gate schemes and to account for two-qubit gate errors and memory depolarisation, in addition to entanglement error. We found that the results depended on input state and that cat-comm and 1TP often yield different output fidelities for a given input state, despite each using the same number and type of two-qubit gates, ebits, measurements and classical communications.

Considering error parameters obtainable with current trapped-ion hardware, we found that inter-QPU entanglement error causes easily the most degradation of the output state, followed by two-qubit gate errors and then memory depolarisation, which has a relatively negligible impact over the time scale of a remote gate. The high magnitude of entanglement error relative to the other error types in current hardware, causes it dominate. However, if high-quality entanglement could be generated between QPUs, to the extent that the magnitudes of gate and entanglement errors are the same, then we find that gate errors become more impactful than inter-QPU entanglement errors. This is most likely due to the greater number of intra-QPU gates than inter-QPU ones.

Once we had explored the behaviour of individual remote gates, which are the building blocks of distributed quantum circuits, we then extended our analysis to 22 larger quantum circuits sampled from the MQT Bench benchmarking suite [4]. We found the same qualitative trends that were observed for individual remote gates also held for the circuits considered, aside from some observed saturation in the larger quantum circuits when the output fidelity reached its minimum value.

We also discussed the feasibility of near-term demonstrations of quantum data centres. We found that if current commercial QPUs and experimentally observed entangling links could be successfully integrated to form a single quantum data centre, a circuit containing around 10 remote gates could be implemented with cat-comm before the output fidelity fell as low as 50%. Improving the entanglement error until its magnitude matched that of errors in local two-qubit gates allowed at least 30 remote gates to be implemented before the fidelity fell to 50%. Although, the number of remote gates that can be carried out could most likely be improved with more optimal compilation strategies, our results suggested that some form of error handling would be needed to deal with entanglement error before quantum data centres could compete with single-processor devices.

To meet this need, we introduced the notion of localised quantum error detection in remote gates, as a half-way house to full fault-tolerance, and considered conventional entanglement distillation techniques. We found that for the hardware parameters considered, entanglement distillation performs best. It improves the output fidelity better or comparably over the entire parameter range considered while using fewer qubits than the quantum error detection setups considered and

unlike the quantum error detection, entanglement distillation does not require discarding any unknown quantum information from the processing qubits if it fails. That said quantum error detection does yield significant improvements over the unencoded case too.

Of the entanglement distillation schemes considered, four-ebit, two-round DEJMPS unsurprisingly mitigates error far more than two-ebit, one-round DEJMPS and probably at least two rounds of entanglement distillation will be required for entanglement distillation to be worthwhile. That said, there are trade offs between the much better output fidelity improvement provided by four-ebit DEJMPS and the latency cost of doing it relative to the two-ebit case. Nonetheless, our work demonstrates that significant improvements to the cost of implementing remote gates on quantum data centres can be obtained with a relatively modest resource cost.

7.1 Future outlook and partially addressed questions

As is typical of research, along with the answers this thesis provides come many questions. Time being the finite resource that it is means that certain questions just could not be answered within the period set out for this PhD project. Some questions were looked into but time or resource constraints prevented their detailed exploration or extension. For other questions, it was immediately understood that they would have to be deferred to the future. We consider both types of question here.

In Sec. 7.1.1, we discuss preliminary work on a specific use case of quantum data centres and how quantum data centres are affected by increasing the number of QPUs a given circuit is split between. We also explore how this work could be extended in the future. Then, in Sec. 7.2, we consider questions raised by the thesis as a whole that merit future investigation.

7.1.1 Resource states and the impact of QPU number

The work in this thesis has focussed in large part on two-QPU scenarios because two QPUs is the minimum required to investigate distributed behaviour, and so two-QPU systems represent a natural starting point for learning about quantum data centres. However, it is natural to ask what happens when the number of QPUs is increased.

Another natural question is what should our quantum data centre actually be used for? This thesis has predominantly assumed that the quantum data centre is being used for running arbitrary quantum circuits specified by a single user but a variety of other paradigms exist. One such use case is measurement-based quantum computing (MBQC) in which a known resource state is created and local measurements, whose bases depend on preceding measurement results, are conducted on it [134]. Here, we consider paradigms in which a quantum data centre is used as a server to generate such resource states for a client who could then specify what measurements to conduct. The advantage of such a set up would be a potentially simplified optimisation and verification process because we can focus our efforts on generating a known resource state whose behaviour and imperfections can be well understood. To this end, we investigate the impact of increasing the number of QPUs used when generating known resource states.

Three resource states are considered. The first, the generalised GHZ state, has the form $\frac{1}{\sqrt{2}}(|000...\rangle + |111...\rangle)$ and is generated by the circuit shown in Fig. 7.1(a). The second is a linear cluster state, while the third is a 2D square array cluster state; see Figs. 7.1(b) and 7.1(c), respectively. A cluster state is created by first initialising qubits in the $|+\rangle$ state and then doing controlled-Z (CZ) gates between the qubits. Cluster states are perhaps easiest to represent as graphs with the vertices being the qubits initialised in the $|+\rangle$ state and the edges representing the subsequent CZ gates. In all cases, the nine-qubit version of the state is considered.

The 2D cluster state, 2DC, is most commonly used for measurement-based quantum computing but the GHZ state can be useful for quantum communications [135], clock synchronisation [136] and metrology tasks [137, 138]. The linear cluster state, LC, has a very similar quantum circuit structure to the GHZ state, albeit with CNOTs swapped for CZ gates and Hadamards applied to every qubit, rather

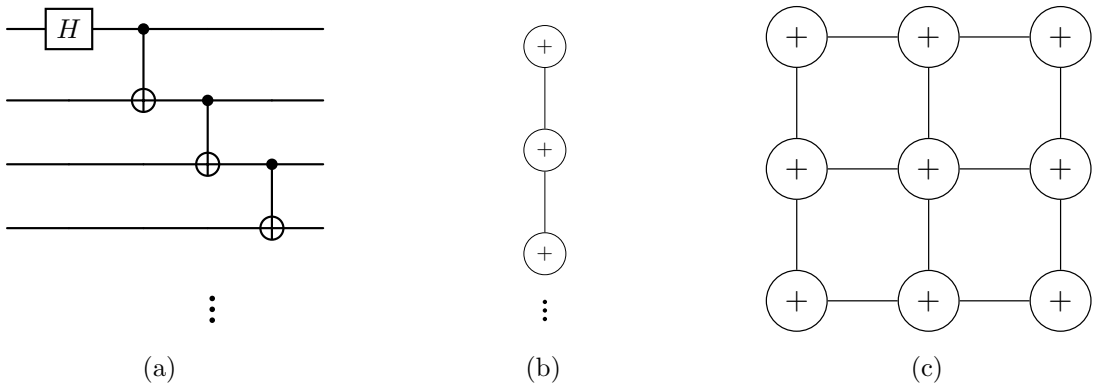
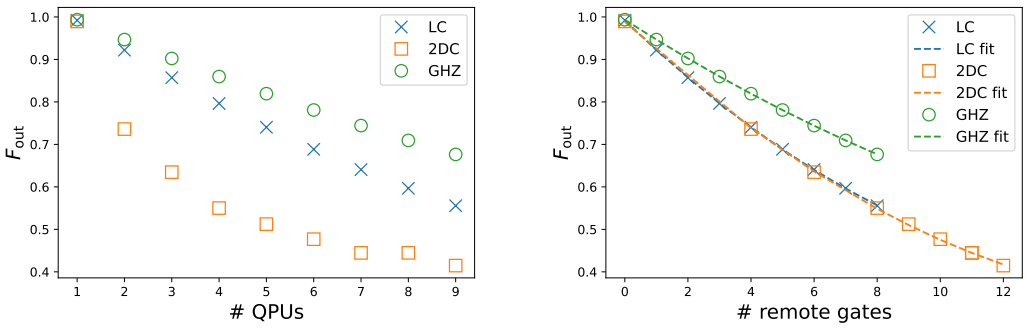


Figure 7.1: The resource states considered: (a) the generalised GHZ state, (b) the linear cluster state and (c) the 2D square array cluster state. (a) is the circuit diagram used to generate the GHZ state, while (b) and (c) show a graph representation of the state with each vertex being a qubit initialised in the $|+\rangle$ state and the edges being a CZ implemented on the vertices it joins after the qubits have been initialised in the $|+\rangle$ state. Although, we consider the nine-qubit implementations of each of the states, we show only four of the qubits for the GHZ state and three of the qubits for the linear cluster state, for clarity. The former is extended to nine-qubits by adding more CNOT gates in the same pattern and the latter is generalised by adding more vertices and edges vertically.

than just the first, and so is included to reveal sensitivities to the exact operations used.

The same noise models and simulation tools utilised in previous chapters are again used here, although, we use the notation ϵ_{tg} instead of ϵ_{cnot} to refer to two-qubit gate errors because CZ gates are now used in the cluster state generation circuits and so not all two-qubit gates are CNOT gates anymore. The noise parameters initially considered are $F_{\text{w}} = 0.94$, $\epsilon_{\text{sg}} = 1.8 \times 10^{-5}$, $\epsilon_{\text{tg}} = 9.7 \times 10^{-4}$, $\epsilon_{\text{m}} = 2.33 \times 10^{-3}$ and $r = 0.055\text{Hz}$, which are the same as those used in Chapter 6 except that F_{w} is fixed at 0.94, the value obtained experimentally in Ref. [109]. The figure of merit is the output fidelity, F_{out}.

We generate each of the considered resource states in turn, splitting the nine qubits used in each resource state between an increasing number of QPUs. Qubits are allocated as evenly as possible to each QPU, and any remainder, after an initial even allocation of qubits to all QPUs, is dealt with by allocating one qubit at a time to each QPU until no more qubits remain. For example, for the four-QPU case, each QPU is initially assigned two qubits each and then the, arbitrarily



(a) (b)

Figure 7.2: The output fidelity as a function of: (a) the number of QPUs and (b) the number of remote gates. In both cases, the parameters $F_{\text{w}} = 0.94$, $\epsilon_{\text{sg}} = 1.8 \times 10^{-5}$, $\epsilon_{\text{tg}} = 9.7 \times 10^{-4}$, $\epsilon_{\text{m}} = 2.33 \times 10^{-3}$ and $r = 0.055$Hz are used.

assigned, first QPU is given an extra qubit, so that all nine qubits are allocated to a QPU. We show F_{out} as a function of the number of QPUs in Fig. 7.2(a).

From Fig. 7.2(a), we can see that the GHZ state is the least vulnerable to increasing the number of QPUs. LC performs worse as the number of QPUs is increased and 2DC unsurprisingly performs the worst, due to having the most two-qubit gates. Given the similarities in their circuit structure, it is interesting that the GHZ state and LC differ as much as they do, with the discrepancy reaching as high as $\frac{(F_{\text{out}})_{\text{GHZ}} - (F_{\text{out}})_{\text{LC}}}{(F_{\text{out}})_{\text{LC}}} \times 100 = 21.7\%$.

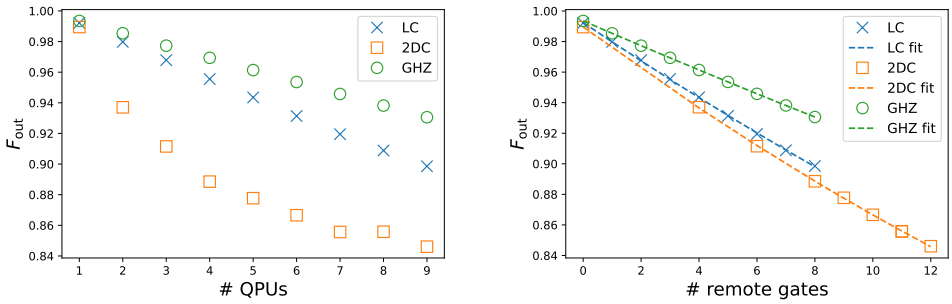
One may also note the more complicated curve exhibited by 2DC relative to the other states but this is simply due to the relationship between the number of QPUs and the number of remote gates used in the resource state generation circuits. To see this, we consider F_{out} as a function of the number of remote gates, x, required when partitioning the resource state generation circuits between a given number of QPUs. The results are shown in Fig. 7.2(b) along with quadratic fits of the form $ax^2 + bx + c$ for real numbers a, b and c.

From Fig. 7.2(b), we observe that our quadratic fits agree excellently with the data. For completeness, the fitting parameters used, which are obtained using the `curve_fit` function from Python's SciPy library [139], are summarised in Tab. 7.1.

Fig. 7.2(b) also gives us some insight into the feasibility of using QDCs for

State	a	b	c
GHZ	9.73×10^{-4}	-4.74×10^{-2}	0.993
LC	2.10×10^{-3}	-7.12×10^{-2}	0.991
2DC	1.85×10^{-3}	-6.97×10^{-2}	0.988

Table 7.1: The fitting parameters used to fit the data shown in Fig. 7.2(b) to a quadratic expression with the form $ax^2 + bx + c$ for real numbers a, b and c. All numbers are quoted to three significant figures.



(a) (b)

Figure 7.3: The output fidelity as a function of: (a) the number of QPUs and (b) the number of remote gates with F_{w} improved beyond the current experimental regime. In both cases, the parameters $F_{\text{w}} = 0.999$, $\epsilon_{\text{sg}} = 1.8 \times 10^{-5}$, $\epsilon_{\text{tg}} = 9.7 \times 10^{-4}$, $\epsilon_{\text{m}} = 2.33 \times 10^{-3}$ and $r = 0.055$Hz are used.

resource state generation. We consider the percentage difference between the QDC and monolithic cases, $\Delta_F = \frac{(F_{\text{out}})_{\text{mono}} - (F_{\text{out}})_{\text{QDC}}}{(F_{\text{out}})_{\text{mono}}} \times 100$, for a given number of qubits. We find that the minimum cost of distributing the 2DC circuit, which is the one most useful for MBQC, is $\Delta_F = 25.6\%$, which occurs when two QPUs and four remote gates are used. When nine QPUs and twelve remote gates are used, $\Delta_F = 58.1\%$. For the GHZ state, the cost of using a QDC is more manageable. We find that $\Delta_F = 4.70\%$ for a two-QPU QDC and $\Delta_F = 31.9\%$ for a nine-qubit QDC.

We also consider the scenario in which F_{w} is improved to 0.999 to make it similar to the two-qubit gate error, $\epsilon_{\text{tg}} = 9.7 \times 10^{-4}$. The full results for this are shown in Fig. 7.3. For 2DC, we find that $\Delta_F = 5.30\%$ for a two-QPU QDC and $\Delta_F = 14.5\%$ for a 9-qubit QDC. For GHZ, $\Delta_F = 0.813\%$ for a two-QPU QDC

and $\Delta_F = 6.33\%$ for a nine-QPU QDC. We also note that a slight separation in the F_{out} values of the LC and 2DC cases as a function of the number of remote gates becomes discernible with these parameters, due to the greater number of local operations.

Overall, it seems likely that any demonstrations of QDCs as a way of generating 2DC cluster states for MBQC will require error handling. Such demonstrations become far more feasible when error handling is used, although the cost relative the monolithic case remains notable. By contrast, it may be practical to generate the GHZ state for various communication tasks using QDCs without error handling and with error handling this becomes extremely efficient. In either case, the output fidelity falls off rapidly with the number of remote gates used. Empirically, we find this decrease to be quadratic, for the specific nine-qubit constructions that we consider. As such, minimising the number of remote gates is key.

This work indicates that QDCs with appropriate error handling may have a role to play within larger networks for tasks like resource state generation to be used as a cloud resource. However, a lot of questions remain about the details of this usage. For example, the details of the client-server relationship have not been explored and neither has the impact of performing the measurements. Moreover, there is space to consider more formal verification techniques beyond consideration of the output fidelity. It would also be interesting to consider more specific use cases and benchmark the performance of MBQC against the gate-based model for the matter-based quantum data centres with photonic interconnects implicitly considered here.

Other potential avenues for future exploration include the application of QDC-based MBQC for blind quantum computing [140], in which security rather than computational power is the key motivation. It may be that there is little benefit to the QDCs in such a scenario but this does merit further thought. It would also be interesting to consider further scaling of the resource states and QDCs considered to larger systems, which, due to the Clifford nature of the gates considered, should be possible to simulate efficiently using stabilisers.

7.2 Questions for future work

As well as further exploring the use of quantum data centres for resource state generation and considering other specific use cases for quantum data centres, a variety of other extensions to the work presented in this thesis exist.

One recurring theme that has cropped up during this thesis is that the input state to a remote gate plays a role in the quality of the output. It would be potentially informative to explore this in more detail and include consideration of arbitrary, potentially entangled, states in the analysis. Understanding the role of input state, would potentially enable the type of remote gate used to be chosen more effectively during compilation, allowing a more optimised selection between cat-comm and TP-comm to be made.

Another avenue for further exploration would be the use of more hardware-specific quantum connections between QPUs. For example, we could use noise models specific to a given qubit modality and ebit distribution scheme, such as trapped ions entangled via photons using a central Bell state measurement [141], and we could also expand our focus from quantum data centres to more hardware-specific multi-core architectures [51], such as spin [142] or superconducting qubits [143] coupled using shared resonators. The simulation tools developed during our work allow for a great deal of detail to be provided in the hardware specification and so it would be interesting to compare and contrast different hardware or architectural choices.

Finally, it would be useful to extend our work into the fault-tolerant regime. We have explored error handling of remote gates as a stepping stone towards fault-tolerance but it would be interesting to see how well the considered techniques can be integrated with a full fault-tolerant architecture. Some in-roads to this have been made through the theoretical consideration of surface code codes on a distributed quantum computer [104, 120, 144–146], however, one of the advantages of the quantum data centre architecture is the ability to scale smaller matter-based QPUs which typically have far greater qubit connectivities than the surface code assumes and so more resource efficient codes are likely to be feasible in the quantum data centre context. Some recent work has also considered the performance of hyberbolic floquet codes in a distributed quantum computing context [147] but

that work does not consider integrating entanglement distillation or other error mitigation techniques into their code to make quantum error correction easier and there remain many possible quantum error correction codes to explore in the distributed quantum computing context. If quantum data centres are truly to scale up quantum computers until they can be used for useful applications then demonstrating a practical advantage over monolithic devices in the fault-tolerant regime is critical.

Appendix A

Analytical derivations of the output fidelity for 1TP and cat-comm

In the main text, the working for Eqs. (4.5) and (4.7), the analytical expressions of F_{out} for 1TP and cat-comm, respectively, are omitted for brevity. Here, we provide the working for both expressions.

For both calculations, we assume that the only source of noise is entanglement error, which we model by assuming all ebits are in the Werner state given by Eq. (4.1). The Werner state is a classical mixture of the four Bell states, $\{|\Phi^{\pm}\rangle = \frac{1}{\sqrt{2}}(|00\rangle \pm |11\rangle), |\Psi^{\pm}\rangle = \frac{1}{\sqrt{2}}(|01\rangle \pm |10\rangle)\}$. Therefore, it is instructive to consider what happens to 1TP and cat-comm when each of the Bell states is distributed between QPUs.

The 1TP scheme depicted in Fig. 4.1(b) assumes that the $|\Phi^{+}\rangle$ state, specifically, was distributed between QPUs during the teleportation process. If one of the other Bell states is distributed instead then a Pauli error occurs in the teleported state. We represent this error as a fictitious gate $R_i \in \{\mathbb{1}, X, Z, ZX\}$, where X and Z are the Pauli X and Z operators, respectively.

Further simplifications can be made using the fact that when any of the Bell states is distributed between QPUs, the output is independent of the BSM results used during the teleportation process, provided that there are no local, intra-QPU, errors of any kind. This is simply verified by calculating the output

A. ANALYTICAL DERIVATIONS OF THE OUTPUT FIDELITY FOR 1TP AND CAT-COMM

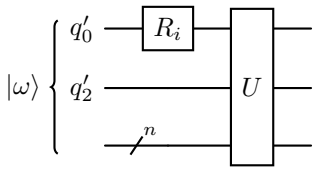


Figure A.1: A simplified version of the 1TP quantum circuit shown in Fig. 4.1(b) after the teleportation has occurred. The circuit shown is derived from Fig. 4.1(b) by assuming a measurement result of '0' for all measurements in the circuit and generalising to an arbitrary remote gate, rather than a remote CNOT gate, and that the teleported qubit is initially separable from the other qubits acted on by U. $|\omega\rangle$ is the ideal state produced after an ideal teleportation from QPU A to QPU B. n is an arbitrary, non-negative number of qubits, on QPU B from Fig. 4.1(b). R_i is a rotation of the ideal teleported state imposed by the possible non-ideality of the entangled state distributed, and U is a quantum gate local to QPU B's qubits. The qubit labels q_0' and q_2' refer to the same qubits as in the original circuit shown in Fig. 4.1(b).

for each measurement result and can be intuitively understood by noting that any measurement result of qubits on QPU A in Fig. 4.1(b) is equally likely to be obtained, regardless of which Bell state is distributed between QPU A and QPU B. Consequently, the circuit shown in Fig. 4.1(b) can be greatly simplified by assuming a specific BSM result has occurred, without loss of generality. For simplicity, we assume here that the BSM result is '00', and obtain a simplified circuit diagram for implementing an arbitrary remote gate using 1TP, as shown in Fig. A.1.

Combining the results for each of the Bell states, the overall output state, ρ_{out}, can be represented in terms of the ideal input state, ρ_ω, as

$$\rho_{\text{out}} = U(\sum_{i=1}^{4} p_i R_i \rho_\omega R_i^\dagger)U^\dagger, \tag{A.1}$$

where p_i is the probability of being in one of the Bell states (corresponding to the coefficients in the Werner state), and U is a unitary gate acting on QPU B from Fig. A.1. For example, if a remote CNOT gate were being considered, as in Fig. 4.1(b), U would be the X (or NOT) gate. Here, we consider the more general case in which U is arbitrary and the remote gate need not be a controlled-unitary (CU) gate.

Making the further, limiting, assumption that the processing qubits involved in the remote gate are initially pure and separable from one another, we obtain:

$$\begin{aligned}\rho_{\text{out}} &= U \sum_{i=1}^{4} p_i R_i \left|\omega\right\rangle \left\langle\omega\right| R_i^\dagger U^\dagger \\ &= \sum_{i=1}^{4} p_i |U R_i \left|\omega\right\rangle|^2\end{aligned} \tag{A.2}$$

where $|\omega\rangle$ is the ideal input obtained when $|\Phi^+\rangle$ is distributed between QPU A and QPU B. The environment, E, may include qubits within the quantum data centre and fictitious environmental qubits external to it.

Combining Eq. (A.2) with the definition of fidelity relative to an ideal, pure state, Eq. (2.6), we find

$$\begin{aligned}F_{\text{out}} &= \sum_i p_i |\left\langle\omega\right| U^\dagger U R_i \left|\omega\right\rangle|^2 \\ &= \sum_i p_i |\left\langle\omega\right| R_i \left|\omega\right\rangle|^2,\end{aligned} \tag{A.3}$$

where in the last line we have used the fact that $U^\dagger U = \mathbb{1}$.

From there, we consider each of the four summands in turn. We use the fact that $|\omega\rangle$ is a pure state and can be expressed as $\alpha|0\rangle + \beta|1\rangle$, where α and β are complex numbers and are subject to the condition that $|\alpha|^2 + |\beta|^2 = 1$. We also note that $p_0 = F_{\text{w}}$ and $p_1 = p_2 = p_3 = \frac{1-F_{\text{w}}}{3}$.

For $i = 0$, the summand is

$$F_{\text{w}}(\alpha\left|0\right\rangle + \beta\left|1\right\rangle)(\alpha^\dagger\left\langle 0\right| + \beta^\dagger\left\langle 1\right|). \tag{A.4}$$

For $i = 1$, the summand is

$$\begin{aligned}&\frac{1-F_{\text{w}}}{3}(\alpha\left|0\right\rangle + \beta\left|1\right\rangle)X(\alpha^\dagger\left\langle 0\right| + \beta^\dagger\left\langle 1\right|) \\ &= \frac{1-F_{\text{w}}}{3}(\alpha\left|0\right\rangle + \beta\left|1\right\rangle)(\alpha^\dagger\left\langle 1\right| + \beta^\dagger\left\langle 0\right|).\end{aligned} \tag{A.5}$$

For $i = 2$, the summand is

$$\begin{aligned}&\frac{1-F_{\text{w}}}{3}(\alpha\left|0\right\rangle + \beta\left|1\right\rangle)Z(\alpha^\dagger\left\langle 0\right| + \beta^\dagger\left\langle 1\right|) \\ &= \frac{1-F_{\text{w}}}{3}(\alpha\left|0\right\rangle + \beta\left|1\right\rangle)(\alpha^\dagger\left\langle 0\right| - \beta^\dagger\left\langle 1\right|).\end{aligned} \tag{A.6}$$

For $i = 3$, the summand is

$$\begin{aligned} &\frac{1-F_{\mathrm{w}}}{3}(\alpha\,|0\rangle + \beta\,|1\rangle)XZ(\alpha^{\dagger}\,\langle 0| + \beta^{\dagger}\,\langle 1|) \\ &= \frac{1-F_{\mathrm{w}}}{3}(\alpha\,|0\rangle + \beta\,|1\rangle)(\alpha^{\dagger}\,\langle 1| - \beta^{\dagger}\,\langle 0|). \end{aligned} \tag{A.7}$$

Summing Eqs. (A.4) to (A.7) and cancelling terms, we find

$$F_{\mathrm{out}} = \frac{1 + F_{\mathrm{w}}(3(|\alpha|^2 + |\beta|^2) - 1)}{3}, \tag{A.8}$$

where we have made frequent use of the orthonormality of the computational basis states $|0\rangle$ and $|1\rangle$.

Using the fact that $|\alpha|^2 + |\beta|^2 = 1$, we obtain the final result

$$F_{\mathrm{out}} = \frac{1 + 2F_{\mathrm{w}}}{3}, \tag{A.9}$$

which is Eq. (4.5) from the main text.

Remarkably, this simple result has no dependence on the local operation, U, applied to QPU B's qubits, provided that U is unitary. The result is also independent of the input state to the processing qubits.

For cat-comm, things are complicated by the measurement on QPU B in the cat-disentanglement subroutine. This means that, although a similar simplified circuit to that depicted in Fig. A.1 can be concocted, the operation U would no longer be unitary and so this way of thinking about the problem is less helpful. Consequently, it is more convenient to use the density matrix formalism throughout the calculation.

The circuit output remains independent of the measurement outcomes, as before, and so there again exist four possible pure output states, one for each of the terms in the Werner state. To make the problem tractable, we again make the limiting assumption that the processing qubits are initially separable. This reduces the generality of the results greatly but does allow some initial investigations to be made and means that simulated results can be checked. With these assumptions made, the possible output states, $|\mathrm{out}\rangle$, to the circuit when the ebits used are in the state $|\Phi^{\pm}\rangle$ or $|\Psi^{\pm}\rangle$ are:

$$|\Phi^{\pm}\rangle_{q_0,q_0'} \rightarrow |\mathrm{out}\rangle_{\mathrm{q_2},\mathrm{q_2'}} = \alpha|0\rangle_{\mathrm{q_2}}|\chi\rangle_{\mathrm{q_2'}} \pm \beta|1\rangle_{\mathrm{q_2}}\mathrm{U}_{\mathrm{q_2'}}|\chi\rangle_{\mathrm{q_2'}}, \tag{A.10}$$

$$|\Psi^{\pm}\rangle_{q_0,q_0'} \rightarrow |\text{out}\rangle_{q_2,q_2'} = \alpha|0\rangle_{q_2} U_{q_2'}|\chi\rangle_{q_2'} \pm \beta|1\rangle_{q_2}|\chi\rangle_{q_2'}, \tag{A.11}$$

where here and hereafter we have highlighted discrepancies between the states in red. $|\chi\rangle_{q_2'}$ is the arbitrary pure state of qubit q_2' from Fig. A.1.

The output states from Eqs. (A.10) and (A.11) correspond to the density matrices:

$$\begin{aligned}\rho_{|\Phi^{\pm}\rangle} = &|\alpha|^2 \,|0\rangle\langle 0|_{q_2} \,|\chi\rangle\langle\chi|_{q_2'} \pm \alpha\beta^* \,|0\rangle\langle 1|_{q_2} \,|\chi\rangle\langle\chi|_{q_2'} \, U_{q_2'}^{\dagger} \\ &\pm \beta\alpha^* \,|1\rangle\langle 0|_{q_2} \, U_{q_2'} \,|\chi\rangle\langle\chi|_{q_2'} \\ &+ |\beta|^2 \,|1\rangle\langle 1|_{q_2} \, U_{q_2'} \,|\chi\rangle\langle\chi|_{q_2'} \, U_{q_2'}^{\dagger},\end{aligned} \tag{A.12}$$

and

$$\begin{aligned}\rho_{|\Psi^{\pm}\rangle} = &|\alpha|^2 \,|0\rangle\langle 0|_{q_2} \, U_{q_2'} \,|\chi\rangle\langle\chi|_{q_2'} \, U_{q_2'}^{\dagger} \pm \alpha\beta^* \,|0\rangle\langle 1|_{q_2} \, U_{q_2'} \,|\chi\rangle\langle\chi|_{q_2'} \\ &\pm \beta\alpha^* \,|1\rangle\langle 0|_{q_2} \,|\chi\rangle\langle\chi|_{q_2'} \, U_{q_2'}^{\dagger} + |\beta|^2 \,|1\rangle\langle 1|_{q_2} \,|\chi\rangle\langle\chi|_{q_2'} \,,\end{aligned} \tag{A.13}$$

respectively.

Moreover, the ideal input state, $|\omega\rangle$, from before now has the state

$$|\omega\rangle = (\alpha\,|0\rangle + \beta\,|1\rangle)_{q_2}(a\,|0\rangle + b\,|1\rangle)_{q_2'}, \tag{A.14}$$

where a and b are complex numbers such that $|a|^2 + |b|^2 = 1$.

Inserting Eq. (A.14) into Eq. (2.6) gives

$$\begin{aligned}F_{\text{out}} &= \sum_{i=1}^{4} p_i \,\langle\omega|_{q_2,q_2'} \,(\rho_i)_{q_2,q_2'} \,|\omega\rangle_{q_2,q_2'} \\ &= \sum_{i=1}^{4} p_i(|\alpha|^2 \,\langle\chi|_{q_2'} \,\langle 0|_{q_2} \,(\rho_i)_{q_2,q_2'} \,|0\rangle_{q_2} \,|\chi\rangle_{q_2'} \\ &\quad + \alpha^*\beta \,\langle\chi|_{q_2'} \,\langle 0|_{q_2} \,(\rho_i)_{q_2,q_2'} \,|1\rangle_{q_2} \, U_{q_2'} \,|\chi\rangle_{q_2'} \\ &\quad + \beta^*\alpha \,\langle\chi|_{q_2'} \, U_{q_2'}^{\dagger} \,\langle 1|_{q_2} \,(\rho_i)_{q_2,q_2'} \,|0\rangle_{q_2} \,|\chi\rangle_{q_2'} \\ &\quad + |\beta|^2 \,\langle\chi|_{q_2'} \, U_{q_2'}^{\dagger} \,\langle 1|_{q_2} \,(\rho_i)_{q_2,q_2'} \,|1\rangle_{q_2} \,|\chi\rangle_{q_2'})\end{aligned} \tag{A.15}$$

with ρ_i being the density matrix for one of the states in Eqs. (A.12) to (A.13).

Inserting (A.12) and (A.13) into (A.15) for each i, and using the orthonormality

of $\{|0\rangle, |1\rangle\}$ to cancel terms yields

$$\begin{aligned} & p_{1,2} \langle\omega|_{q_2,q_2'} (\sigma_{1,2})_{q_2,q_2'} |\omega\rangle_{q_2,q_2'} \\ = & \Big(|\alpha|^4 \left|\langle\chi|\chi\rangle_{q_2'}\right|^2 \pm |\alpha|^2|\beta|^2 \langle\chi|\chi\rangle_{q_2'} \langle\chi| U_{q_2'}^\dagger U_{q_2'} |\chi\rangle_{q_2'} \\ & \pm |\alpha|^2|\beta|^2 \langle\chi| U_{q_2'}^\dagger U_{q_2'} |\chi\rangle_{q_2'} \langle\chi|\chi\rangle_{q_2'} \\ & + |\beta|^4 \left|\langle\chi|_{q_2'} U_{q_2'}^\dagger U_{q_2'} |\chi\rangle_{q_2'}\right|^2 \Big) p_{1,2}, \end{aligned} \tag{A.16}$$

for $i = 1$ and $i = 2$, where $p_{1,2} = p_1$ or p_2.

Using $UU^+ = U^+U = \mathbb{1}$ and $\langle\chi|\chi\rangle = 1$, then this simplifies to

$$\begin{aligned} & \left(|\alpha|^4 \pm 2|\alpha|^2|\beta|^2 + |\beta|^4\right) p_{1,2} \\ & = \left(|\alpha|^2 \pm |\beta|^2\right)^2 p_{1,2}. \end{aligned} \tag{A.17}$$

For $i = 1$ (+ case), the expression further simplifies to $p_1 = F_\mathrm{w}$ because $|\alpha|^2 + |\beta|^2 = 1$, by the normalisation condition of quantum states.

Proceeding in a similar vein, for $i = 3$ and $i = 4$:

$$\begin{aligned} & p_{3,4} \langle\omega|_{q_2,q_2'} (\sigma_{3,4})_{q_2,q_2'} |\omega\rangle_{q_2,q_2'} \\ = & |\alpha|^4 \left|\langle\chi| U_{q_2'}^\dagger |\chi\rangle_{q_2'}\right|^2 \pm |\alpha|^2|\beta|^2 \langle\chi| U_{q_2'} |\chi\rangle_{q_2'} \\ & \pm |\alpha|^2|\beta|^2 \left(\langle\chi|_{q_2'} U_{q_2'}^\dagger |\chi\rangle_{q_2'}\right)^2 \\ & + |\beta|^4 \left(\langle\chi|_{q_2'} U_{q_2'}^\dagger |\chi\rangle_{q_2'}\right)^2. \end{aligned} \tag{A.18}$$

Inserting these terms back into Eq. (A.15), we find that

$$\begin{aligned} F_\mathrm{out} = F_\mathrm{w} + \frac{1 - F_\mathrm{w}}{3} (& \left(|\alpha|^2 - |\beta|^2\right)^2 + 2|\alpha|^4 \left|\langle\chi| U |\chi\rangle_{q_2'}\right|^2 \\ & + 2|\beta|^4 \left(\langle\chi| U^\dagger |\chi\rangle_{q_2'}\right)^2), \end{aligned} \tag{A.19}$$

which is equivalent to Eq. (4.6) in the main text. Unlike the expression for 1TP, Eq. (A.19) depends on the exact local operations conducted on QPU B's qubits and the input state of QPU A and QPU B's processing qubits.

Appendix B

Variation with respect to input state

Throughout Sec. 4.5 in the main text, we assume that the input state to the remote CNOT gate considered is given by Eq. (4.10). In this appendix, we consider the implications of this assumption and directly investigate the role played by input state.

In Sec. B.1, we demonstrate that the output error after a remote CNOT gate depends on its input, discuss the implications of our choice of input state and indicate why cat-comm and 1TP differ. In Sec. B.2 we verify that our main results from Chapter 4 are robust to variations in input state and comment on a specific observation from Chapter 4 that is sensitive to the choice of input state.

B.1 The impact of input state on remote CNOT gates

For 1TP, it is clear from Eq. (4.5) that the output fidelity, and therefore output error, is unaffected by the input to the remote gate when only entanglement error is considered. However, the same cannot be said for cat-comm, 2TP, and TP-safe. Moreover, for cat-comm, some input state dependence can be seen in the analytical expression for output state fidelity given by Eq. (4.6). To fully understand the role of input state would require a non-trivial averaging over all

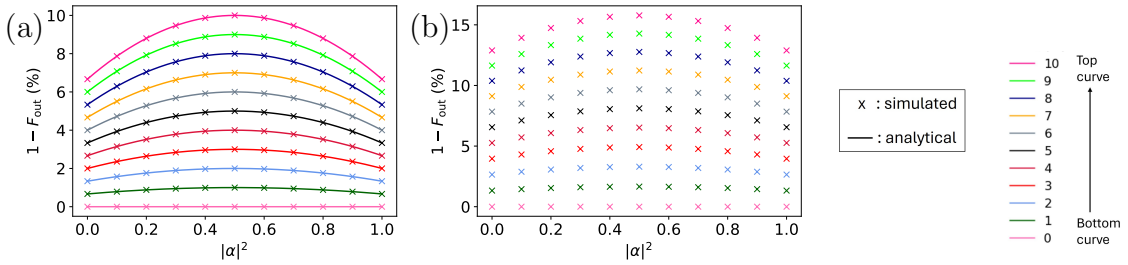


Figure B.1: Output fidelity as a function of input state for a remote CNOT implemented using: (a) cat-comm, (b) 2TP. The input state of the control qubit for the remote CNOT gate is changed by varying $|\alpha|^2$, where α is the coefficient from Eq. (B.1). Markers represent simulated data and solid lines represent analytical data created using Eq. (4.7).

possible quantum states for the control and target qubits, which correspond to q_2 and q_2' respectively in Fig. 4.1. Such an averaging would most likely require a detailed analytical description of the problem and multivariate integration or Monte Carlo analysis. This is beyond the scope of Chapter 4.

Instead, for convenience, we constrain the problem by assuming that the input state has the form:

$$|\text{input}\rangle_{q_2,q_2'} = (\alpha\,|0\rangle_{q_2} + \beta\,|1\rangle_{q_2})\,|0\rangle_{q_2'}\,, \tag{B.1}$$

where q_2 and q_2' are the processing qubits depicted in Fig. 4.1; $|\text{input}\rangle_{q_2,q_2'}$ is the input state of qubits q_2 and q_2', prior to the remote CNOT gate taking place; and α and β are complex numbers such that $|\alpha|^2 + |\beta|^2 = 1$.

To identify the input state, with the form of Eq. (B.1), that gives the lowest output fidelity, in Fig. B.1, we show the output error as a function of the input state of q_2. As noted in the main text, 2TP and TP-safe give identical results when only entanglement error is considered, and so only cat-comm and 2TP are considered in Fig. B.1.

From Fig. B.1, it is apparent that there is a clear maximum in the output error when $|\alpha|^2 = \frac{1}{2}$. This means that the magnitude squared of the coefficients $\alpha = \beta = \frac{1}{\sqrt{2}}$ in Eq. (4.10) correspond to the highest output error of any input state of the form given by Eq. (B.1).

The output error is independent of the relative phase between α and β. For cat-comm, the phase independence of the output error can be seen from Eq. 4.6,

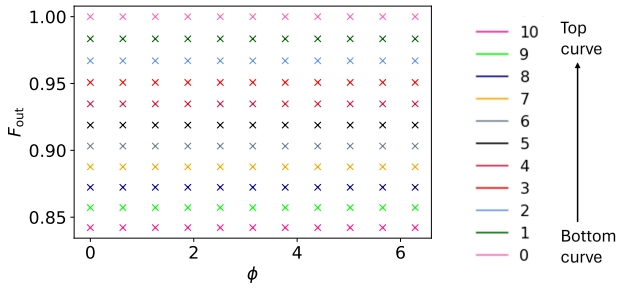


Figure B.2: Simulated output fidelity as a function of the relative phase, ϕ, in the input state for a remote CNOT implemented using 2TP. ϕ is defined as in Eq. (B.2). $\alpha = \frac{1}{\sqrt{2}}$, where α is the coefficient from Eq. (B.2).

which has no terms that depend on the relative phases of α and β. For 2TP, we can see the phase independence of the output error by assuming, without loss of generality, that α is real and re-writing Eq. (B.1) as:

$$|\text{input}\rangle_{q_2,q_2'} = (\alpha\,|0\rangle_{q_2} + e^{i\phi}\sqrt{1-|\alpha|^2}\,|1\rangle_{q_2})\,|0\rangle_{q_2'}\,, \tag{B.2}$$

where $0 \leq \phi < 2\pi$ is a real number. We plot the simulated output error as a function of ϕ in Fig. B.2. The flatness of the curves in Fig. B.2, indicates that the results are independent of the phase, ϕ.

The independence of the output error to phase terms in the input indicates that any input state with $|\alpha|^2 = \frac{1}{2}$ can yield the maximum output error for input states with the form given by Eq. (B.1). We choose $\alpha = \beta = \frac{1}{\sqrt{2}}$ for convenience, recovering Eq. (4.10).

To get some insight into the ramifications of our choice of input state, we consider the slightly more general case where only entanglement error is varied and the input state of q_2 and q_2' is separable. In such a scenario, the input state of q_2 and q_2' is given by Eq. (4.4), which can be re-written, up to a global phase, more explicitly as:

$$\begin{aligned}|\text{input}\rangle_{q_2,q_2'} =&(\alpha\,|0\rangle_{q_2} + e^{i\phi}\sqrt{1-|\alpha|^2}\,|1\rangle_{q_2})\otimes \\ &(\gamma\,|0\rangle_{q_2'} + e^{i\theta}\sqrt{1-|\gamma|^2}\,|1\rangle_{q_2'}),\end{aligned} \tag{B.3}$$

where α, γ are real numbers with magnitudes $0 \leq |\alpha| \leq 1$ and $0 \leq |\gamma| \leq 1$, and $0 \leq \phi < 2\pi$ and $0 \leq \theta < 2\pi$ are real numbers.

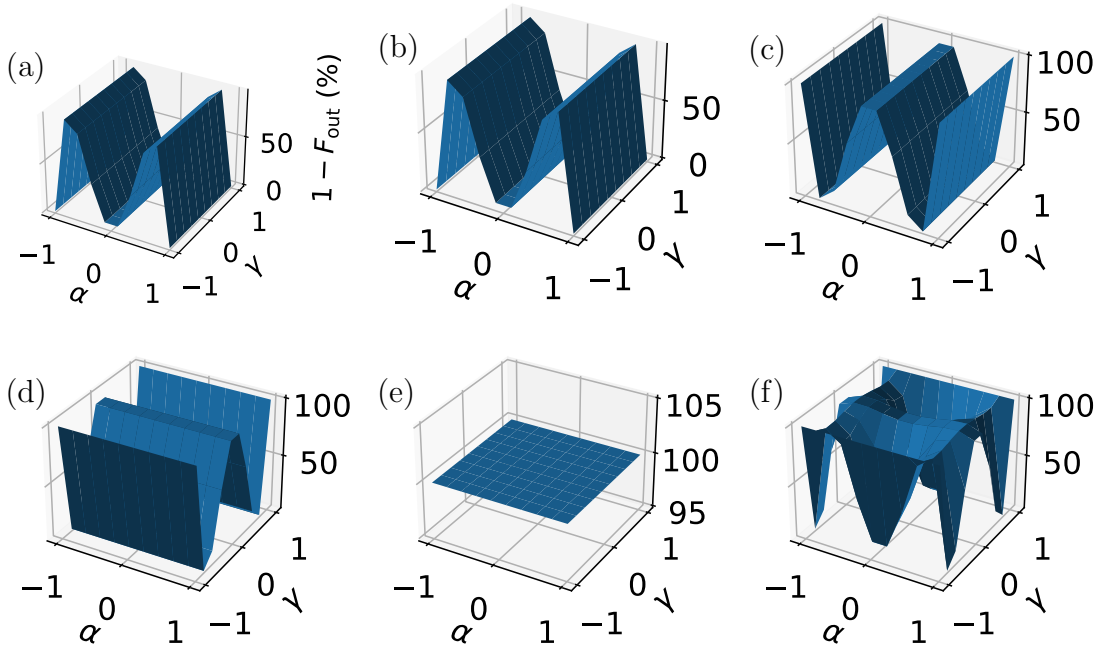


Figure B.3: The output error between the state outputted from a remote CNOT gate when: $|\Phi^+\rangle$ is distributed as the ebit and when: (a)-(b) $|\Phi^-\rangle$ is distributed as the ebit for 1TP and cat-comm, respectively; (c)-(d) $|\Psi^+\rangle$ is distributed as the ebit for 1TP and cat-comm, respectively; and when (e)-(f) $|\Psi^-\rangle$ is distributed as the ebit for 1TP and cat-comm, respectively. All other forms of error are set to zero. α and γ are the coefficients from Eq. (B.3).

We model the entanglement error by assuming that ebits are in the Werner state, see Eq. (4.1), rather than $|\Phi^+\rangle$, as discussed in Sec. 4.3.2. As the Werner state is just a mixture of the Bell states, we separately consider the output error when each of the Bell states is distributed as an ebit. Figures B.3-B.4 show surface plots of the output error between the state outputted from a remote CNOT gate when the ideal state, $|\Phi^+\rangle$, is distributed as an ebit and the state outputted when one of the other Bell states is erroneously distributed as an ebit for different values of α and γ, and ϕ and θ, respectively.

Several observations can be made from Figs. B.3-B.4:

1. Eq. (4.10) is at or near a local maximum in output error for cat-comm. Figures B.3(b), B.3(f), which show the output error with varying α and γ for cat-comm, have maxima at approximately $\alpha = \frac{1}{\sqrt{2}} \approx 0.707$, while Fig.

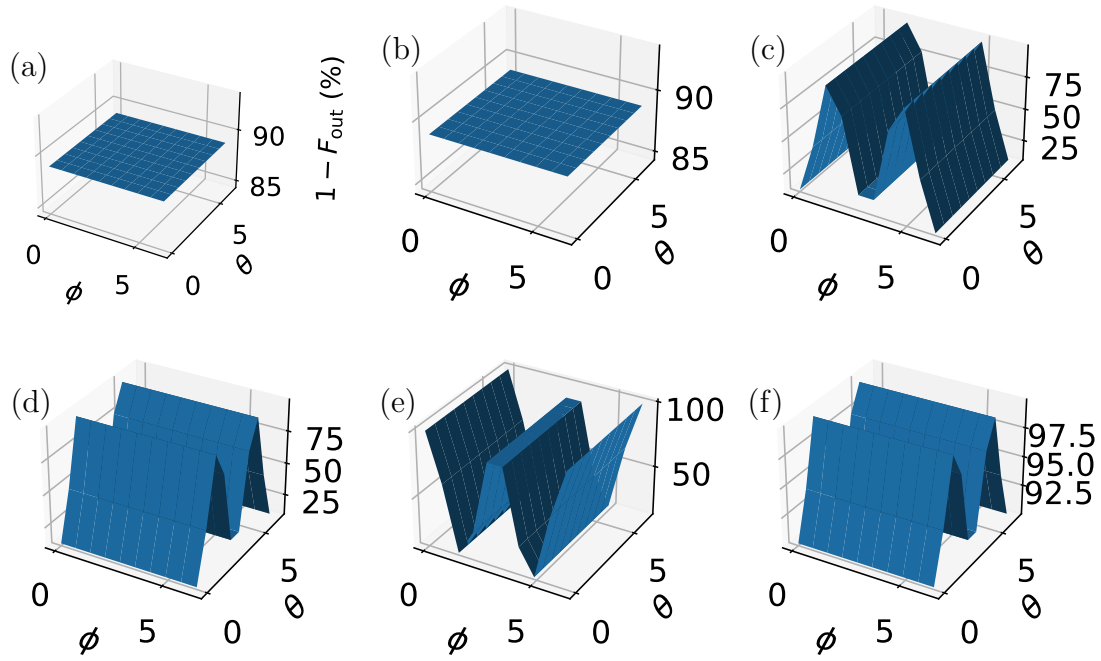


Figure B.4: The output error between the state outputted from a remote CNOT gate when $|\Phi^+\rangle$ is distributed as the ebit and when: (a)-(b) $|\Phi^-\rangle$ is distributed for 1TP and cat-comm, respectively; (c)-(d) $|\Psi^+\rangle$ is distributed for 1TP and cat-comm, respectively; and (e)-(f) $|\Psi^-\rangle$ is distributed for 1TP and cat-comm, respectively. All other forms of error are set to zero. We fix $\alpha = \gamma = \frac{1}{\sqrt{3}}$ and vary ϕ and θ, the relative phases of the input state for q_2 and q_2', respectively, as defined in Eq. (B.3).

B.3(d), which is also for cat-comm, is constant with respect to α but has maxima at $\gamma = \pm 1$, corresponding to the state $|0\rangle$. Therefore, although the choice of Eq. (4.10) is somewhat arbitrary, it does correspond to an especially high entanglement error. That said, Eq. (4.10) cannot be said to be a true upper bound on the entanglement error without knowing the global maximum.

2. Cat-comm and 1TP differ due to discrepancies in the circuit output produced by the $|\Psi^{\pm}\rangle$ terms only. The $|\Phi^{\pm}\rangle$ terms yield identical outputs for cat-comm and 1TP.

3. Cat-comm is constant with respect to ϕ but varies with respect to θ. This can be seen from Figs. B.4(b), B.4(d) and B.4(f).

4. In the presence of entanglement error only, 1TP gives a constant output error for any input state, but cat-comm does not. This could be seen from Eqs. (4.5) and (4.6), but the reason behind it is made clearer by Fig. B.3 and Fig. B.4. For 1TP, the different output errors resemble out of phase oscillations. When the output error caused by the distribution of one Bell state is low, it is high for a different Bell state. For example, Fig. B.3(a) has maxima at the points where Fig. B.3 (c) has minima and slopes up at the α values where Fig. B.3 (c) slopes down. The only other term, shown in Fig. B.3 (e) is constant with respect to α and γ and so does not disrupt this trend. Similar observations can be made about phase by comparing Fig. B.4(a), Fig. B.4(c), and Fig. B.4(e). By contrast, for cat-comm the output errors caused by the distribution of each Bell state are often rotated by $\frac{\pi}{2}$ radians relative to each other, as in Figs. B.3(b) and B.3(d), and Figs. B.4(d) and B.4(f). This means that the different error terms do not balance out in the same way as they do for 1TP. This accounts for the difference between cat-comm and 1TP.

B.2 Verification that results from Ch. 4 are robust to input state

While observation 1 from Sec. B.1 indicates that Eq. (4.10) corresponds to an input state of particular interest, it is also clear from Sec. B.1 that cat-comm, 2TP, and, by extension, TP-safe will vary with respect to input state. As such, it is important to understand if certain claims made in the main text are robust to input state variation. In particular, it is not clear from the figures shown in the main text alone that observation i. from Sec. 4.5.1 and any of the observations made in Sec. 4.5.3 hold for arbitrary input states. Although a true proof is not possible without more detailed input state analysis, here we demonstrate that the claims from the main text hold for a variety of input states. In Sec. B.2.1, we demonstrate that observation i. from Sec. 4.5.1 is robust to variation in input state and make some further comments on the impact of input state variation on the relative difference between first-order and simulated output errors values. In Sec. B.2.2, we verify the observations about the relative impact of different error types made in Sec. 4.5.3 of the main text. Finally, in Sec. B.2.3, we make some further comments on the observations made about the relative impact of different remote gate schemes in Sec. 4.5.2 of the main text.

B.2.1 Verification of observations from Sec. 4.5.1

To verify observation i. from Sec. 4.5.1 and corroborate comments made about the input state dependence of observation iv., we consider the percentage difference, Δ_{oe}, calculated using Eq. (4.11), between the output error computed using the first-order approximation and the simulator, respectively, for various different input states. This is shown in Fig. B.5. Figure B.5(a) shows the collated results for a variety of input states when ϵ_{ebit} and ϵ_{cnot}, respectively, are varied with all other errors set to zero. The input states considered all have the form given by Eq. (B.3) and parameter values produced using a permutation of the parameters $\alpha \in \{0.2, \frac{1}{\sqrt{2}}\}$, $\gamma = \{0.0, 0.2, ..., 1.0\}$, $\phi = 0$, $\theta \in \{0, \frac{2\pi}{5}, ..., 2\pi\}$. We average over all of the input states that can be produced in this way. In Fig. B.5(b), the input state $|\text{input}\rangle_{q_2, q_2'} = (0.2\,|0\rangle_{q_2} + 0.980\,|1\rangle_{q_2}) \otimes (0.6\,|0\rangle_{q_2'} + 0.8\,|1\rangle_{q_2'})$ [1] is considered in

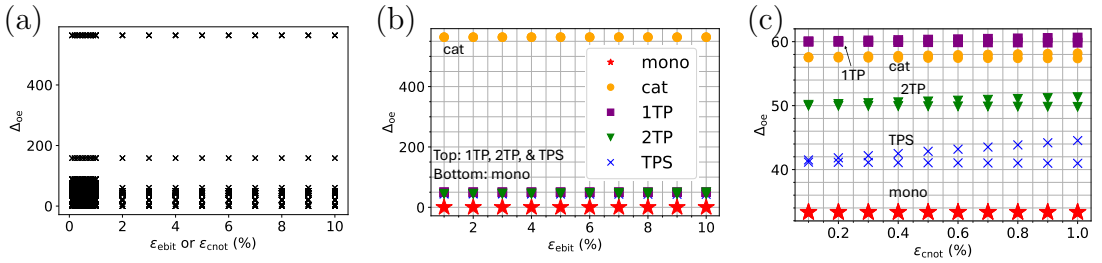


Figure B.5: The percentage difference, Δ_{oe}, in the output error calculated using the first-order approximations, given by Eqs. (4.8) and (4.9), and the simulator, respectively. In (a), we show the collated results for a variety of input states when ϵ_{ebit} and ϵ_{cnot}, respectively are varied with all other errors set to zero. The input states considered have the form given by Eq. (B.3) and are produced from all permutations of $\alpha \in \{0.2, \frac{1}{\sqrt{2}}\}$, $\gamma = \{0.0, 0.2, ..., 1.0\}$, $\phi = 0$, $\theta \in \{0, \frac{2\pi}{5}, \ldots, 2\pi\}$. Only Eq. (4.9) and the simulated results are compared—Eq (4.8) is not considered. In (b), the only non-zero error parameter is ϵ_{ebit} and the input state parameters are: $\alpha = 0.2$, $\gamma = 0.6$, $\phi = 0$, $\theta = 2\pi$. In (c), the only non-zero error parameter is ϵ_{cnot} and the input state parameters are: $\alpha = \frac{1}{\sqrt{2}}$, $\gamma = 0.8$, $\phi = 0$, $\theta = 2\pi$. Whenever two curves appear with the same markers, the top curve uses (4.8) for the first-order approximation and the bottom curve uses (4.9). In all cases, Δ_{oe} is calculated using Eq. (4.11).

the presence of a varying non-zero ϵ_{ebit} with all other errors set to zero. Similarly, in Fig. B.5 (c), $|\text{input}\rangle_{q_2, q_2'} = \frac{1}{\sqrt{2}}(|0\rangle_{q_2} + |1\rangle_{q_2}) \otimes (0.8\,|0\rangle_{q_2'} + 0.600\,|1\rangle_{q_2'})$ [1] is considered in the presence of a varying non-zero ϵ_{cnot} with all other errors set to zero.

Figure B.5(a) corroborates observation i. from Sec. 4.5.1, which states that the first order approxmations loosely upper bound the simulated output error. All percentage differences, Δ_{oe}, in Fig. B.5(a) are non-negative and so for all of the input states considered, the approximate results are greater than or equal to the exact results. Figures B.5(b)-(c) differ both qualitatively and quantitatively from Fig. 4.2(a) and Fig. 4.2(c), with a different ordering of the Δ_{oe} values for the various remote gate schemes and different quantitative values of Δ_{oe}. This indicates that the agreement between the output error predicted by the first-order approximations and the simulation are highly input state dependent, as suggested in observation iv. from Sec. 4.5.1.

[1] All rounded coefficients are displayed here to three significant figures but floating point precision is used for the generation of data.

B.2.2 Verification of all observations from Sec. 4.5.3

The next observations to verify are observations i. and ii. from Sec. 4.5.3. Both observations are centered around the relative impact of the different error types introduced in Sec. 4.3.2 with entanglement error in the state-of-the-art and distilled ranges, respectively. In Fig. B.6, we show the output error when one non-zero error parameter is varied while all other error parameters are fixed at zero. For entanglement error, the state-of-the-art range of values is considered. Much like Fig. 4.4 from the main text, in each subfigure, we show the output errors generated by three different error types for a given remote gate scheme, but here, we average each data point over the output errors obtained at a given error parameter value for a variety of input states. Again, the input states considered have the form given by Eq. (B.3) and are produced from all permutations of $\alpha \in \{0.2, \frac{1}{\sqrt{2}}\}$, $\gamma = \{0.0, 0.2, ..., 1.0\}$, $\phi = 0$, $\theta \in \{0, \frac{2\pi}{5}, ..., 2\pi\}$. Averaging is done by taking the mean, in Figs. B.6(a), B.6(c), B.6(e), B.6(g), B.6(i) and the median, in Figs. B.6(b), B.6(d), B.6(f), B.6(h), and B.6(i). When the mean is used, the error bars show the standard deviation and when the median is used, the error bars show the interquartile range. Figure B.7 shows the same thing but with entanglement error varied over the distilled range.

Figures B.6 and B.7 indicate that observations i. and ii. from Sec. 4.5.3 do indeed hold for a variety of input states. The output error caused by each error type in Fig. B.6, in which the state-of-the-art parameters are used, is consistently smallest for memory depolarisation. Local gate error yields a larger output error but its impact is in turn dominated by entanglement error. Again memory depolarisation is relatively negligible. These findings hold true for all remote gate schemes and are true for both the mean and median over input state with neither the standard deviation nor the interquartile range of any data points overlapping—which would have indicated uncertainty in the relative impact of the different error types. All of this is consistent with observation i.

Similarly, in Fig. B.7, in which the distilled range of entanglement error is used, the output error is smallest when memory depolarisation is non-zero, greater for entanglement error, and greatest for local gate error. This is consistent with observation ii. from Sec. 4.5.3.

B. VARIATION WITH RESPECT TO INPUT STATE

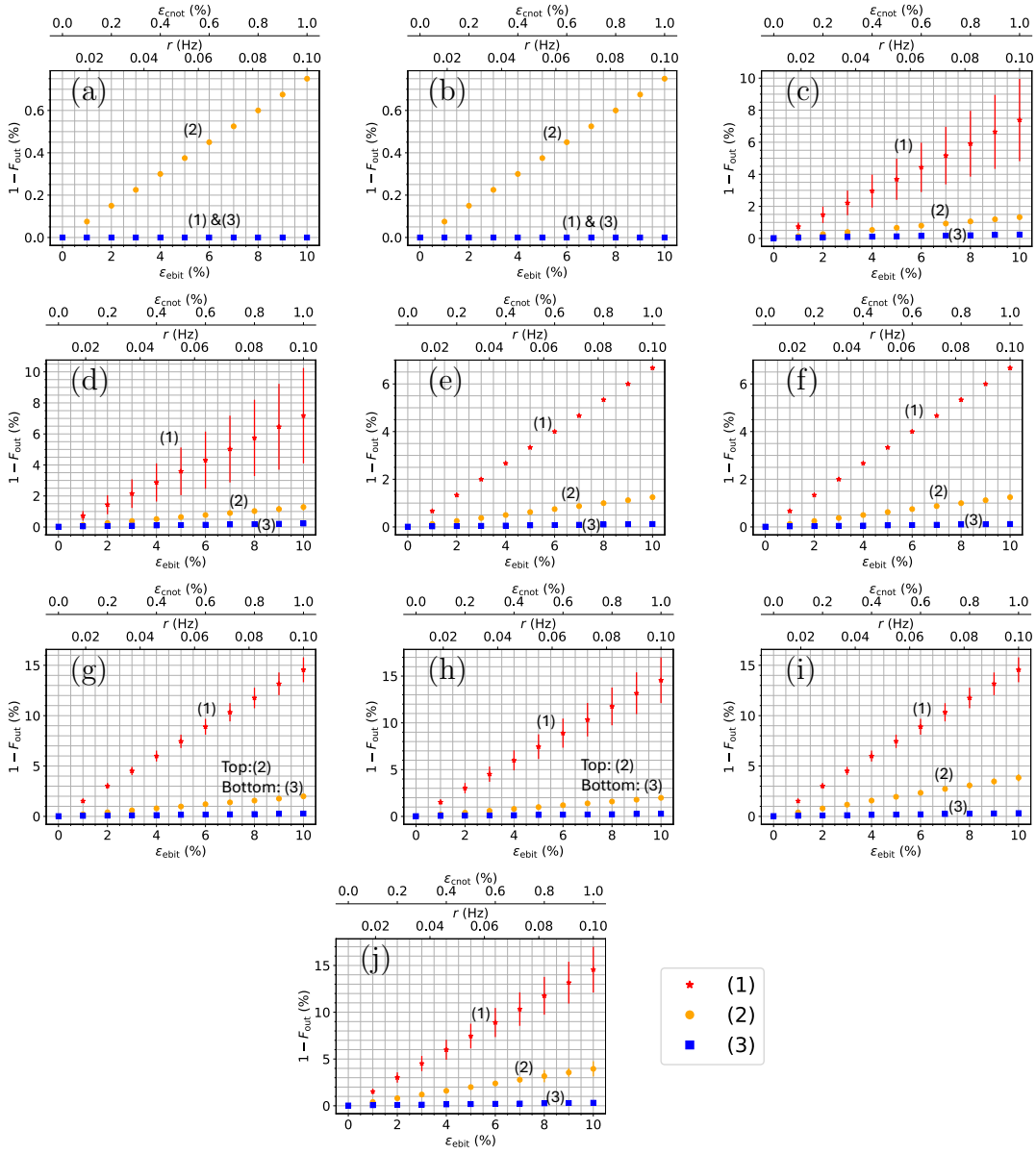


Figure B.6: The output error as a function of (1) ϵ_{ebit} within the state-of-the-art range, (2) ϵ_{cnot}, (3) r, for a single CNOT gate implemented using: (a)-(b) a monolithic processor; (c)-(d) cat-comm; (e)-(f) 1TP; (g)-(h) 2TP; and (i)-(j) TP-safe. The results are averaged over a variety of input states with the form given by Eq. (B.3) and the parameters varied over all permutations of $\alpha \in \{0.2, \frac{1}{\sqrt{2}}\}$, $\gamma = \{0.0, 0.2, ..., 1.0\}$, $\phi = 0$, $\theta \in \{0, \frac{2\pi}{5}, \ldots, 2\pi\}$. The average used is the mean for (a), (c), (e), (g), and (i), with error bars indicating the standard deviation, and the median for (b), (d), (f), (h), and (j) remote gates, with error bars indicating the interquartile range. For curves (1), (2), and (3) on each figure, the non-varied error parameters are set to zero. Due to the low standard error previously observed when averaging over simulation runs, only one simulation run is used for each input state.

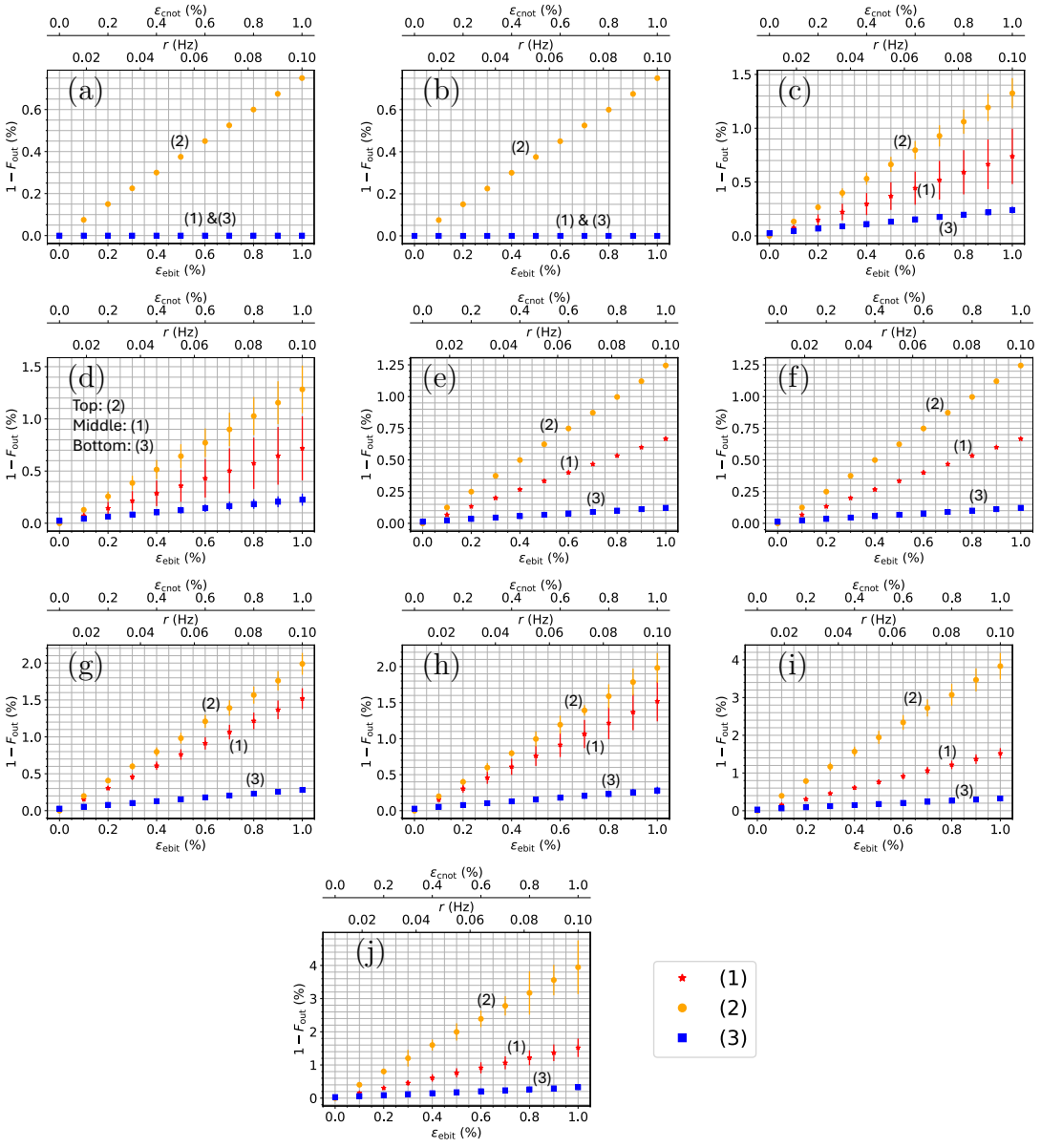


Figure B.7: The output error as a function of (1) ϵ_{ebit} within the distilled range, (2) ϵ_{cnot}, (3) r, for a single CNOT gate implemented using: (a)-(b) a monolithic processor; (c)-(d) cat-comm; (e)-(f) 1TP; (g)-(h) 2TP; and (i)-(j) TP-safe. The results are averaged over a variety of input states with the form given by Eq. (B.3) and the parameters varied over all permutations of $\alpha \in \{0.2, \frac{1}{\sqrt{2}}\}$, $\gamma = \{0.0, 0.2, ..., 1.0\}$, $\phi = 0$, $\theta \in \{0, \frac{2\pi}{5}, ..., 2\pi\}$. The average used is the mean for (a), (c), (e), (g), and (i), with error bars indicating the standard deviation, and the median for (b), (d), (f), (h), and (j), with error bars indicating the interquartile range. For curves (1), (2), and (3) on each figure, the non-varied error parameters are set to zero. Due to the low standard error previously observed when averaging over simulation runs, only one simulation run is used for each input state.

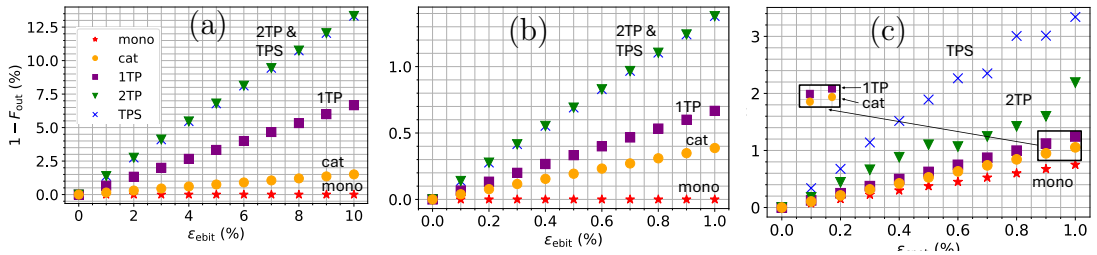


Figure B.8: The output error, $1 - F_{\text{out}}$, for an individual remote CNOT gate with non-zero: (a) entanglement error in the state-of-art range and an input state of $|\text{input}\rangle_{q_2,\, q_2'} = (0.2\,|0\rangle_{q_2} + 0.980\,|1\rangle_{q_2}) \otimes (0.6\,|0\rangle_{q_2'} + 0.8\,|1\rangle_{q_2'})$; (b) entanglement error in the distilled range and an input state of $|\text{input}\rangle_{q_2,\, q_2'} = (0.2\,|0\rangle_{q_2} + 0.980\,|1\rangle_{q_2}) \otimes (0.4\,|0\rangle_{q_2'} + 0.917\,|1\rangle_{q_2'})$; (c) local two-qubit gate error and an input state of $|\text{input}\rangle_{q_2,\, q_2'} = (0.2\,|0\rangle_{q_2} + 0.980\,|1\rangle_{q_2}) \otimes (0.8\,|0\rangle_{q_2'} + 0.6\,|1\rangle_{q_2'})$. The output errors when the remote CNOT gate is implemented using cat-comm (cat), 1TP, 2TP, and TP-safe (TPS) are considered. For the monolithic case (mono) a single local CNOT gate is considered. Due to the low standard error previously observed when averaging over simulation runs, only one simulation run is used for each input state.

The robustness of observations i. and ii. from Sec. 4.5.3 can be further verified by an exhaustive search of plots showing the non-averaged output errors for each input state individually. We perform such a search and find that the ordering seen in observations i. and ii. is seen consistently for all of input states with the form of Eq. (B.3) and parameters $\alpha \in \{0.2, \frac{1}{\sqrt{2}}\}$, $\gamma = \{0.0, 0.2, ..., 1.0\}$, $\phi = 0$, $\theta \in \{0, \frac{2\pi}{5}, \ldots, 2\pi\}$. The relevant plots can be found in the supplementary information.

B.2.3 Comments on Sec. 4.5.2

In Sec. 4.5.2 from the main text, we note that the ordering of remote gate schemes for the input state given by Eq. (4.10), is, ordered from lowest to highest by their impact on output error: 1TP, cat-comm, 2TP, and then TP-safe. 2TP and TP-safe are equally damaging to the output when only entanglement error is considered. Here, we confirm that the relative positions of 1TP and cat-comm are specific to input state, as alluded to in the main text and Sec. B.1.

Figure B.8 shows a comparison of the output error caused by each of the different remote gate schemes shown in Fig. 4.1. Unlike in the main text, the results for variety of different input states are shown. In Figs. B.8(a)-(b),

the entanglement error is varied over the state-of-the-art and distilled ranges, respectively, and all other errors are set to zero. In Fig. B.8(c), the local two-qubit gate error is varied and all other errors are set to zero.

In Figs. B.8(a)-(c), 1TP has a larger output error than cat-comm, in contrast to what is seen and discussed in Sec. 4.5.2 from the main text, in which a different input state, given by Eq. (4.10), is used. This is indicative of how input state dependent the relative ordering of the schemes can be. Interestingly, when we consider only a non-zero memory depolarisation, we have not yet found an input state for which 1TP does not outperform cat-comm. The most probable reason for this is that cat-comm has a slightly higher latency than 1TP. 1TP communicates the results of both the measurements that occur in the scheme at the same point in the circuit. This means that both measurements can be combined into a single classical message at the same time. By contrast, for cat-comm, the measurements are split between the cat-entanglement and cat-disentanglement protocols, which must occur sequentially. This means that the measurement results must be communicated between the QPUs in separate messages at different times and so the portion of the latency attributable to classical messaging is doubled. The remaining remote gate schemes retain the order quoted in the main text.

Appendix C

Analytical calculations of success probability

The success probability, p_{s}, of 4QED/SS is provided in Eq. (6.9) of the main text. Here, we derive Eq. (6.9).

We assume that only entanglement errors are present and that all ebits used during teleportation have the form given in Eq. (4.1). After teleportation of the encoded qubit, but prior to decoding, we obtain the state

$$\rho_{\mathrm{t}} = \sum_{i=1}^{4}\sum_{j=1}^{4}\sum_{k=1}^{4}\sum_{l=1}^{4} p_{ijkl} R_{ijkl} \rho_{\mathrm{ideal}} R_{ijkl}^{\dagger}, \tag{C.1}$$

where $R_{ijkl} = R_i \otimes R_j \otimes R_k \otimes R_l$ for $R_i \in \{\mathbb{1}, X, Z, XZ\}$ and $i, j, k, l \in \{1, 2, 3, 4\}$, p_{ijkl} is the probability of error R_{ijkl} occurring, and ρ_{ideal} is the ideal teleported state.

Of all the summands, only those for which no error occurs, or errors occur but remain undetected, count towards the success probability. Only one term in Eq. (C.1) has no errors and this occurs with probability

$$p_0 = F_{\mathrm{w}}^4. \tag{C.2}$$

Undetected errors occur when there are an even number of the same type of errors, meaning that the stabiliser measurements remain unchanged from the no error case. This means that there are either two or four different errors on different qubits.

For the two-error case, there are 18 ways in which this can occur, each occurring with probability

$$F_{\mathrm{w}}^2\left(\frac{1-F_{\mathrm{w}}}{3}\right)^2.$$

As such the probability of having exactly two errors, p_2, is

$$p_2 = 18 \times F_{\mathrm{w}}^2\left(\frac{1-F_{\mathrm{w}}}{3}\right)^2 = 2F_{\mathrm{w}}^2(1-F_{\mathrm{w}})^2. \tag{C.3}$$

A similar process yields a four error probability, p_4, of

$$p_4 = \frac{(1-F_{\mathrm{w}})^4}{27}. \tag{C.4}$$

Adding the right-hand sides of Eqs. (C.2), (C.3), and (C.4) will give us Eq. 6.9 in the main text.

Appendix D

Input states and the repetition code performance

In the second observation on Fig. 6.6(a) that we make in Sec. 6.5 of Chapter 6, we note that neither FCRC nor PCRC shows any benefit on average over the unencoded case, but there is a significant advantage to using FCRC/PCRC for some input states and a disadvantage in other cases. Here, we expand upon that statement.

Figure D.1 shows the minimum, maximum and mean, values of F_{out} for a remote CNOT gate implemented using FCRC and PCRC. The parameters are $F_{\text{w}} \in [0.90, 0.99]$, $\epsilon_{\text{sg}} = \epsilon_{\text{tg}} = \epsilon_{\text{m}} = r = 0$. For each datapoint, the result is sampled from 40,000 raw data points for which no error was detected. From Fig. D.1, we can see that for some input states, both FCRC and PCRC correct almost all of the error. However, for other input states, the encoded case actually does worse than the unencoded case, even in the absence of local errors. In such scenarios, it seems that the additional phase errors imposed on the system by using additional imperfect ebits, counteracts any benefit to the repetition code error detection, which detects only bit flips. These effects balance out on average to confer no benefit relative to the unencoded case.

The natural next question to ask is for which input states is F_{out} better and worse than the unencoded case but there does not seem to be a discernible pattern to this behaviour and so we can only conclude that the results vary greatly with the input state.

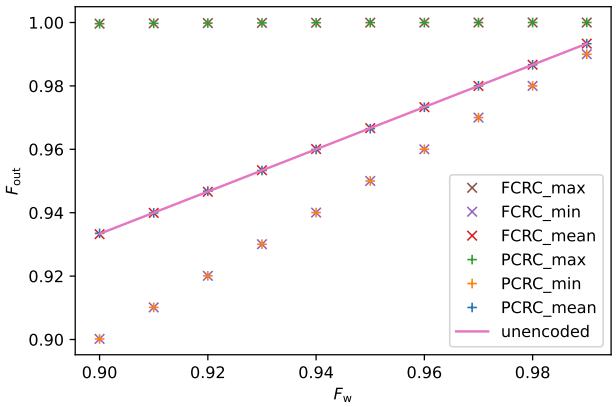


Figure D.1: The min, max and mean F_{out} with respect to the parameters $F_{\text{w}} \in [0.90, 0.99]$, $\epsilon_{\text{sg}} = \epsilon_{\text{tg}} = \epsilon_{\text{m}} = r = 0$, for a remote CNOT gate encoded using FCRC and PCRC.

References

[1] OMG Standards Development Organisation. Unified markup language (uml). https://www.omg.org/uml/. xii, 26, 44

[2] A Yimsiriwattana and S Lomonaco. Generalized GHZ states and distributed quantum computing. *Coding Theory and Quantum Computing*, 381:131–147, 2005. xiii, 6, 54, 58, 71

[3] A Yimisiriwattana and S Lomonaco Jr. Distributed quantum computing: A distributed Shor alogorithm. arXiv:quant-ph/0403146v2, 2004. xiii, 4, 54, 58, 71

[4] N Quetschlich, L Burgholzer, and R Wille. MQT bench: Benchmarking software and design automation tools for quantum computing. *Quantum*, 7: 1062, 2023. xiv, xv, 34, 40, 78, 80, 82, 85, 110

[5] AW Leung, MA Nielsen, IL Chuang, and Y Yamamoto. Approximate quantum error correction can lead to better codes. *Physical Review A*, 56: 2567–2573, October 1997. xvi, xviii, 93, 94, 103, 107

[6] J Roffe. Quantum error correction: An introductory guide. *Contemporary Physics*, 60:226–245, 2019. xvi, 21, 55, 63, 93, 94

[7] CH Bennett, G Brassard, S Popescu, B Schumacher, JA Smolin, and WK Wootters. Purification of noisy entanglement and faithful teleportation via noisy channels. *Physical Review Letters*, 76:722–725, 1996. xviii, 17, 61, 63, 94, 96, 97, 103

[8] D Deutsch, A Ekert, R Jozsa, C Macchiavello, S Popescu, and A Sanpera. Quantum privacy amplification and the security of quantum cryptography over noisy channels. *Physical Review Letters*, 77:2818–2821, 1996. xviii, 17, 63, 94, 97, 100, 103

[9] C Zalka. Simulating quantum systems on a quantum computer. In *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, volume 454, pages 313–322, 1998. 1

[10] P Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM journal on Computing*, 26: 1484–1509, 1997. 1, 3, 79

[11] L Grover. A fast quantum-mechanical algorithm for database search. In *Proceedings of the Twenty-Eigth Annual ACM Symposium on Theory of Computing*, pages 212–219, 1996. 1, 79

[12] R Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21:467–488, 1982. 1

[13] R Feynman. Quantum mechanical computers. *Foundations of Physics*, 16: 507–531, 1986.

[14] D Deutsch. Quantum theory, the church-turing principle and the universal quantum computer. *Proceedings of the Royal Society of London A*, 400: 97–117, 1985.

[15] D Deutsch. Quantum computational networks. *Proceedings of the Royal Society of London A*, 425:73–90, 1989. 1

[16] C Bruzewicz. Trapped-ion quantum computing: Progress and challenges. *Applied Physics Reviews*, 6:021314, 2019. 1

[17] Sanskriti Joshi and Sajjad Moazeni. Scaling up Superconducting Quantum Computers With Cryogenic RF-Photonics. *Journal of Lightwave Technology*, 42(1):166–175, January 2024. ISSN 1558-2213. doi: 10.1109/JLT.2023.3311806. URL https://ieeexplore.ieee.org/document/10239333. 1

[18] L Magnoni. Modern messaging for distributed systems. *Journal of Physics: Conference Series*, 608:012038, 2015. 1

[19] WK Wooters and WH Zurek. A single quantum cannot be cloned. *Nature*, 299:802–803, 1982. 2

[20] H Barnum, C M Caves, C A Fuchs, R Jozsa, and B Schumacher. Noncommuting mixed states cannot be broadcast. *Physical Review Letters*, 76: 2818–2921, 1996. 2

[21] A Wu, H Zhang, G Li, A Shabani, Y Xie, and Y Ding. Autocomm: A framework for enabling efficient communication in distributed quantum programs. In *55th IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 1027–1041, 2022. 2, 35, 54, 55, 67, 68, 71, 101, 102

[22] N Sangouard, C Simon, H de Riedmatten, and N Gisin. Quantum repeaters based on atomic ensembles and linear optics. *Reviews of Modern Physics*, 83:33–80, 2011. 2, 17, 18

[23] K Azuma, SE Economou, D Elkouss, P Hilaire, L Jiang, H Lo, and I Tzitrin. Quantum repeaters: From quantum networks to the quantum internet. *Reviews of Modern Physics*, 95:045006, 2023. 2

[24] P Benioff. The computer as a physical system: A microscopic quantum mechanical hamiltonian model of computers as represented by turing machines. *Journal of Statistical Physics*, 22:563–591, 1980. 3

[25] Y Manin. Computable and uncomputable (in russian). Sovetskoye Radio, Moscow, 1980. 3

[26] R Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21:467–488, 1982. 3

[27] D Deutsch and R Jozsa. Rapid solution of problems by quantum computation. *Proceedings of the Royal Society of London Series A*, 439:553–558, 1992. 3

[28] E Bernstein and U Vazirani. Quantum complexity theory. *SIAM Journal on Computing*, 26:1411–1473, 1997.

[29] D Simon. On the power of quantum computation. *SIAM Journal on Computing*, 26:116–123, 1997. 3

[30] PW Shor. Scheme for reducing decoherence in quantum computer memory. *Physical Review A*, 52:R2493–R2496, 1995. 3, 94

[31] E Knill, R Laflamme, and L Viola. Theory of quantum error correction for general noise. *Physical Review Letters*, 84:2525–2528, 2000. 3, 20

[32] C Monroe, DM Meekhof, BE King, WM Itano, and DJ Wineland. Demonstration of a fundamental quantum logic gate. *Physical Review Letters*, 75: 4714–4717, Dec 1995. 3

[33] Y Nakamura, YA Pashkin, and JS Tsai. Coherent control of macroscopic quantum states in a single-cooper-pair box. *Nature*, 398:786–788, 1999. 3

[34] J Preskill. Quantum computing in the NISQ era and beyond. *Quantum*, 2: 79, 2018. 3

[35] LK Grover. Quantum telecomputation. arXiv:quant-ph/9704012, 1997. 4

[36] R Cleve and H Buhrman. Substituting quantum entanglement for communication. *Physical Review A*, 56:1201–1204, 1997. 4

[37] JI Cirac, AK Ekert, SF Huelga, and C Macchiavello. Distributed quantum computation over noisy channels. *Physical Review A*, 59, 1999. 4, 55, 56

[38] J Eisert, K Jacobs, P Papadopoulos, and M Plenio. Optimal local implementaion of nonlocal quantum gates. *Physical Review A*, 62:052317, 2000. 4, 54, 71

[39] D Collins, N Linden, and S Popescu. Nonlocal content of quantum operations. *Physical Review A*, 64:032302, 2001. 4

[40] L Jiang, JM Taylor, A Sørensen, and MD Lukin. Distributed quantum computation based on small quantum registers. *Physical Review A*, 76: 062323, 2007. 4

[41] A Serafini, S Mancini, and S Bose. Distributed quantum computation via optical fibers. *Physical Review Letters*, 96:010503, 2006.

[42] YL Lim, A Beige, and LC Kwek. Repeat-until-success linear optics distributed quantum computing. *Physical Review Letters*, 95:030505, 2005.

[43] DKL Oi, SJ Devitt, and LCL Hollenberg. Scalable error correction in distributed ion trap computers. *Physical Review A*, 74:052313, 2006. 4

[44] M Gupta and A Pathak. A scheme for distributed quantum search through simultaneous state transfer mechanism. *Annalen der Physik*, 16(12):791–797, 2007. 4

[45] Y Huang, X Ren, Y Zhang, L Duan, and G Guo. Experimental teleportation of a quantum controlled-not gate. *Physical Review Letters*, 93:240501, 2004. 4

[46] W Gao et al. Teleportation-based realization of an optical quantum two-qubit entangling gate. *Proceedings of the National Academy of Sciences*, 107 (49):20869–20874, 2010. 4

[47] KS Chou et al. Deterministic teleportation of a quantum gate between two logical qubits. *Nature*, 561:568–573, 2018. 4

[48] Y Wan et al. Quantum gate teleportation between separated qubits in a trapped-ion processor. *Science*, 364:875–878, 2019. 4

[49] P Maunz et al. Heralded quantum gate between remote quantum memories. *Physical Refview Letters*, 102:250502, 2009. 4

[50] S Daiss et al. A quantum-logic gate between distant quantum-network modules. *Science*, 371:614–617, 2021. 4

[51] M Caleffi et al. Distributed quantum computing: A survey. *Computer Networks*, 254:110672, 2024. 4, 117

[52] D. Main et al. Distributed quantum computing across an optical network link. arXiv:2407.00835, 2024. 5, 87

[53] K Campbell, A Lawey, and M Razavi. Quantum data centres: a simulation-based comparative noise analysis. *Quantum Science and Technology*, 10: 015052, 2024. 6

[54] DM Greenberger, MA Horne, and A Zeilinger. Going beyond bell's theorem. arXiv:0712.0921, 2007. 6

[55] HJ Briegel and R Raussendorf. Persistent entanglement in arrays of interacting particles. *Physical Refview Letters*, 86:910–913, 2001. 6

[56] M Nielsen and I Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 10th anniversary edition, 2010. 9, 10, 12, 13, 15, 20, 56, 62, 79, 90, 94, 99

[57] J Bell. On the Einstein Podolsky Rosen paradox. *Physics*, 1:195–200, 1964. 11, 15

[58] A Aspect et al. Experimental tests of realistic local theories via Bell's theorem. *Physical Review Letters*, 47:460–463, 1981.

[59] D Mermin. Is the moon there when nobody looks? Reality and quantum theory. *Physics Today*, 38:38–47, 1985. 11, 15

[60] R Jozsa. Fidelity for mixed quantum states. *Journal of Modern Optics*, 41: 2315–2323, 1994. 11, 55

[61] Walter Greiner. *Quantum Mechanics: An Introduction*. Springer, 2001. ISBN 978-3-540-67458-0. 12

[62] J Clauser, M Horne, A Shimony, and R Holt. Proposed experiment to test local-hidden variable theories. *Physical Review Letters*, 23:880–884, 1969. 15

[63] K Brown, J Kim, and C Monroe. Co-designing a scalable quantum computer with trapped atomic ions. *npj Quantum Information*, 2:1–10, 2016. 16

[64] WJ Munro, KA Harrison, AM Stephens, SJ Devitt, and K Nemoto. From quantum multiplexing to high-performance quantum networking. *Nature Photonics*, 4:792–796, 2010. 17, 61, 63

[65] L Jiang et al. Quantum repeater with encoding. *Physical Review A*, 79: 032325, 2009. 17, 92

[66] T Coopmans et al. Netsquid, a discrete-event simulation platform for quantum networks. *Communications Physics 4, 164*, 4(164), 2021. 23, 26, 29, 32, 42, 43, 51, 66, 98, 101

[67] X Wu et al. SeQUeNCe: a customizable discrete-event simulator of quantum networks. *Quantum Science and Technology*, 6:045027, 2021. 23

[68] B Bartlett. A distributed simulation framework for quantum networks and channels. arXiv:1808.07047, 2018.

[69] T Matsuo. Simulation of a dynamic, ruleset-based quantum network. arXiv:1908.10758, 2019.

[70] S Diadamo, J Nötzel, B Zanger, and MM Beşe. QuNetSim: A software framework for quantum networks. *IEEE Transactions on Quantum Engineering*, 2:1–12, 2021. 23

[71] Jorge Vázquez-Pérez, Daniel Expósito-Patiño, Marta Losada, Álvaro Carballido, Andrés Gómez, and Tomás F. Pena. CUNQA: a Distributed Quantum Computing emulator for HPC. arXiv:2511.05209, 2025. 24

[72] R Parekh et al. Quantum algorithms and simulation for parallel and distributed quantum computing. In *Proceedings of Second International Workshop on Quantum Computing Software*, pages 9–19, 2021. 24

[73] S Muralidharan. The simulation of distributed quantum algorithms. arXiv:2402.10745, 2025. 24

[74] D Ferrari and M Amoretti. dqc-executor. https://github.com/qis-unipr/dqc-executor. 24

[75] Sen Zhang, Lingjun Xiong, Yipie Liu, Brian L. Mark, Lei Yang, Zebo Yang, and Weiwen Jiang. An end-to-end distributed quantum circuit simulator. arXiv:2511.19791, 2025. 24

[76] K Campbell. dqc_simulator. https://github.com/km-campbell/dqc_simulator.git. 24, 51, 66, 98, 101

[77] nuqasm2. https://github.com/jwoehr/nuqasm2, 2022. 25, 66

[78] NetSquid-PhysLayer. https://gitlab.com/softwarequtech/netsquid-snippets/netsquid-physlayer.git, . 26, 33

[79] NetSquid website. https://netsquid.org/, . 27, 41

[80] Numpy. https://numpy.org/. 31, 41

[81] Qiskit. https://github.com/Qiskit/qiskit.git. 34

[82] AW Cross, LS Bishop, A Smolin, and JM Gambetta. Open Quantum Assembly Language. arXiv:1707.03429, 2017. 34, 66

[83] pandas. https://pandas.pydata.org/. 42

[84] JdO Filho, Z Papp, R Djapic, and J Oosteveen. Model-based design of self-adapting networked signal processing systems. In *IEEE 7th International Conference on Self-Adaptive and Self-Organizing Systems*, pages 41–50, 2013. 42

[85] J Illiano, M Caleffi, A Manzalini, and AS Cacciapuoti. Quantum internet protocol stack: A comprehensive survey. *Computer Networks*, 213:109092, 2022. 46

[86] A Dahlberg et al. A link layer protocol for quantum networks. In *Proceedings of the ACM Special Interest Group on Data Communication*, pages 159–173, 2019. 46, 47

[87] M Pompili et al. Experimental demonstration of entanglement delivery using a quantum network stack. *npj Quantum Information*, 8:121, 2022. 46, 47

[88] CH Bennett et al. Teleporting an unknown quantum state via dual classical and Einstein-Podolsky-Rosen channels. *Physical Review Letters*, 70:1895–1899, 1993. 54, 90

[89] D Bouwmeester et al. Experimental quantum teleportation. *Nature*, 390: 575–579. 54, 90

[90] D Dadkhah, M Zomorodi, SE Hosseini, P Plawiak, and X Zhou. Reordering and partitioning of distributed quantum circuits. *IEEE ACCESS*, 10:70329–70341, 2022. 54, 71

[91] O Daei, K Navi, and M Zomorodi-Moghadam. Optimized quantum circuit partitioning. *International Journal of Theoretical Physics*, 59:3804–3820, 2020.

[92] J Baker, C Duckering, A Hoover, and F Chong. Time-sliced quantum circuit partitioning for modular architectures. In *17th ACM International Conference on Computing Frontiers*, volume 1, pages 98–107, 2020. 71

[93] P Andres-Martinez and C Heunen. Automated distribution of quantum circuits via hypergraph partitioning. *Physical Review A*, 100:032308, 2019.

[94] R Sundaram, H Gupta, and R Ramakrishnan. Efficient distribution of quantum circuits. In *35th International Symposium on Distributed Computing*, pages 41:1–41:20, 2021. 71

[95] I Ghodsollahee, Z Davarzani, M Zomorodi, P Plawiak, M Houshmand, and M Houshmand. Connectivity matrix model of quantum circuits and its application to distributed quantum circuit optimization. *Quantum Information Processing*, 20, 2021.

[96] M Zomorodi-Moghadam, M Houshmand, and M Houshmand. Optimizing teleportation cost in distributed quantum circuits. *International Journal of Theoretical Physics*, 50:848–861, 2018.

[97] MG Davis, J Chung, D Englund, and R Kettimuthu. Towards distributed quantum computing by qubit and gate graph partitioning techniques. arXiv:2310.03942, 2023.

[98] D Ferrari, A Cacciapuoti, M Amoretti, and M Caleth. Compiler design for distributed quantum computing. *IEEE Transactions on Quantum Engineering*, 2:1–20, 2021. 54, 55, 67

[99] A Wu, Y Ding, and A Li. QuComm: Optimizing collective communication for distributed quantum computing. In *Proceedings of the 56th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO, page 479–493. Association for Computing Machinery, 2023. 71

[100] D Ferrari, S Carretta, and M Amoretti. A modular quantum compilation framework for distributed quantum computing. *IEEE Transactions on Quantum Engineering*, 4:1—-13, 2023. 54, 55

[101] A. Yu. Kitaev. Quantum measurements and the abelian stabilizer problem. arXiv:9511026, 1995. 56

[102] N Ezzell, B Pokharel, L Tewala, G Quiroz, and DA Lidar. Dynamical decoupling for superconducting qubits: A performance survey. *Physical Review Applied*, 20:064027, 2023. 56

[103] NMP Neumann, R van Houte, and T Attema. Imperfect distributed quantum phase estimation. In *Computational Science - ICCS 2020, PT VI*, volume 12142 of *Lecture Notes in Computer Science*, pages 605–615, 2020. 56

[104] Y Li and SC Benjamin. Hierarchical surface code for network quantum computing with modules of arbitrary size. *Physical Review A*, 94:042303, 2016. 61, 117

[105] HJ Briegel, W Dür, JI Cirac, and P Zoller. Quantum repeaters: The role of imperfect local operations in quantum communication. *Physical Review Letters*, 81:5932, 1998. 61, 99

[106] M Razavi. *Fiber-Based Quantum Repeaters*, chapter 24, pages 675–691. John Wiley & Sons, Ltd, 2023. ISBN 9783527837427. 62, 63

[107] JW Pan, C Simon, Č Brukner, and A Zeilinger. Entanglement purification for quantum communication. *Nature*, 410:1067–1070, 2001. 63

[108] S Flannigan et al. Propagation of errors and quantitative quantum simulation with quantum advantage. *Quantum Science and Technology*, 7:045025, 2022. 65

[109] LJ Stephenson et al. High-rate, high-fidelity entanglement of qubits across an elementary quantum network. *Physical Review Letters*, 124:110501, 2020. 68, 101, 102, 113

[110] Ionq aria: Practical performance. https://ionq.com/resources/ionq-aria-practical-performance, 2023. 67, 68, 101, 102, 104

[111] TS Metodi, DD Thaker, AW Cross, FT Chong, and IL Chuang. A quantum logic array microarchitecture: Scalable quantum data movement and computation. arXiv:0509051, 2005. 68, 101, 102

[112] Z Yu and Y Li. Analysis of error propagation in quantum computers. arXiv:2209.01699, 2022. 69

[113] M Sarvaghad-Moghaddam and M Zomorodi. A general protocol for distributed quantum gates. *Quantum Information Processing*, 20:265, 2021. 71

[114] M Luo and H Li. Distributed quantum computation assisted by remote toffoli gate. In *Cloud Computing and Security*, pages 475–485. Springer International Publishing, 2016. ISBN 978-3-319-48671-0.

[115] N Nickerson. *Practical Fault-Tolerant Quantum Computing*. PhD thesis, Imperial College London, 2015. 71

[116] A Montanaro. Quantum algorithms: an overview. *npj Quantum Information*, 2(15023), 2016. 79

[117] A Ambainis, J Kempe, and A Rivosh. Coins make quantum walks faster. arXiv:quant-ph/0402107, 2004. 79

[118] D Grinko et al. Iterative quantum amplitude estimation. *npj Quantum Information*, 7(52), 2021. 79

[119] K Campbell. Supplementary information for "Quantum data centres: a simulation-based comparitive noise analysis (Zenodo)". 10.5281/zenodo.14382385, 2024. 84, 86

[120] J Ramette, J Sinclaier, NP Breuckmann, and V Vuletić. Fault-tolerant connection of error-corrected qubits with noisy links. *npj Quantum Information*, 10:58, 2024. 88, 117

[121] D Gottesman. Quantum fault tolerance in small experiments. arXiv:1610.03507, 2016. 90

[122] J Roffe, D Headley, N Chancellor, D Horsman, and V Kendon. Protecting quantum memories using coherent parity check codes. *Quantum Science and Technology*, 3:035010, 2018.

[123] NM Linke et al. Fault-tolerant quantum error detection. *Science Advances*, 3:1701074, 2017. 94

[124] R Harper and ST Flammia. Fault-tolerant logical gates in the IBM quantum experience. *Physical Review Letters*, 122:080504, 2019.

[125] Is error detection helpful on IBM 5q chips? *Quantum Information and Computation*, 18:949–964, 2018.

[126] E. Knill. Quantum computing with realistically noisy devices. *Nature*, 434: 39—-44, 2005. 90

[127] A Gonzales, D Dilley, B Li, L Jiang, and ZH Saleem. Detecting errors in a quantum network with Pauli checks. *Physical Review A*, 111:052629, 2025. 90

[128] N Isailovic, Y Patel, M Whitney, and J Kubiatowicz. Interconnection networks for scalable quantum computers. In *33rd International Symposium on Computer Architecture (ISCA'06)*, pages 366–377, 2006. 90

[129] O Boncalo and A Amaricai. Reliability analysis of qubit data movement for distributed quantum computation. In *12th Euromicro Conference on*

Digital System Design, Architectures, Methods and Tools, pages 481–487, 2009. 90

[130] V Siddhu, E Winston, DC McKay, and A Javadi-Abhari. Basic distillation with realistic noise. arXiv:2504.06175, 2025. 90

[131] Y Jing, D Alsina, and M Razavi. Quantum key distribution over quantum repeaters with encoding: Using error detection as an effective postselection tool. *Physical Review Applied*, 14:064037, 2020. 94

[132] Quantinuum performance validation. https://docs.quantinuum.com/systems/user_guide/hardware_user_guide/performance_validation.html, 2025. 101, 104

[133] R. Paschotta. Fibers. RP Photonics Encyclopedia, 2006. Available online at https://www.rp-photonics.com/fibers.html. 102

[134] HJ Briegel, DE Browne, W Dür, R Raussendorf, and M Van den Nest. Measurement-based quantum computation. *Nature Physics*, 5:19–26, 2009. 112

[135] M Hillery, V Bužek, and A Berthiaume. Quantum secret sharing. *Physical Review A*, 59:1829–1834, 1999. 112

[136] P Kómár et al. A quantum network of clocks. *Nature Physics*, 10, 2014. 112

[137] C Yin and A Lucas. Heisenberg-limited metrology with perturbing interactions. *Quantum*, 8:1303, 2024. 112

[138] D Leibfried et al. Toward heisenberg-limited spectroscopy with multiparticle entangled states. *Science*, 304(5676):1476–1478, 2004. 112

[139] Scipy v1.9.3. https://scipy.org/. 114

[140] JF Fitzsimons. Private quantum computation: an introduction to blind quantum computing and related protocols. *npj Quantum Information*, 2017. 116

[141] WJ Munro, K Azuma, K Tamaki, and K Nemoto. Inside quantum repeaters. *IEEE Journal of Selected Topics in Quantum Electronics*, 21:78–90, 2015. 117

[142] J Dijkema et al. Cavity-mediated iSWAP oscillations between distant spins. *Nature Physics*, 21:168–174, 2025. 117

[143] M Silanpää, J Park, and R Simmonds. Coherent quantum state storage and transfer between two phase qubits via a resonant cavity. *Nature*, 449: 438–442, 2007. 117

[144] S de Bone, P Möller, CE Bradley, TH Taminiau, and D Elkouss. Thresholds for the distributed surface code in the presence of memory decoherence. *AVS Quantum Science*, 6:033801, 2024. 117

[145] J Kim et al. A fault-tolerant million qubit-scale distributed quantum computer. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, volume 2, page 1–19. Association for Computing Machinery, 2024. ISBN 9798400703850.

[146] Y van Montfort, S de Bone, and D Elkouss. Fault-tolerant structures for measurement-based quantum computation on a network. *Quantum*, 9:1723, 2025. 117

[147] E Sutcliffe, B Jonnadula, C Le Gall, AE Moylett, and CM Westoby. Distributed quantum error correction based on hyperbolic floquet codes. arXiv:2501.14029, 2025. 117