On the quantum performance evaluation of two distributed quantum architectures

Vardoyan, Gayane; Skrzypczyk, Matthew; Wehner, Stephanie

DOI
10.1016/j.peva.2021.102242

Publication date
2021

Document Version
Final published version

Published in
Performance Evaluation

Citation (APA)
Vardoyan, G., Skrzypczyk, M., & Wehner, S. (2021). On the quantum performance evaluation of two distributed quantum architectures. *Performance Evaluation*, *153*, Article 102242. https://doi.org/10.1016/j.peva.2021.102242

Performance Evaluation 153 (2022) 102242

Contents lists available at ScienceDirect

Performance Evaluation

journal homepage: www.elsevier.com/locate/peva

On the quantum performance evaluation of two distributed quantum architectures

Gayane Vardoyan *, Matthew Skrzypczyk, Stephanie Wehner

QuTech and Kavli Institute of Nanoscience, Delft University of Technology, Lorentzweg 1, 2628 CJ Delft, The Netherlands

ARTICLE INFO

Article history:
Available online 13 October 2021

Keywords:
Quantum architecture
Fidelity
Markov chain

ABSTRACT

Distributed quantum applications impose requirements on the quality of the quantum states that they consume. When analyzing architecture implementations of quantum hardware, characterizing this quality forms an important factor in understanding their performance. Fundamental characteristics of quantum hardware lead to inherent tradeoffs between the quality of states and traditional performance metrics such as throughput. Furthermore, any real-world implementation of quantum hardware exhibits time-dependent noise that degrades the quality of quantum states over time. Here, we study the performance of two possible architectures for interfacing a quantum processor with a quantum network. The first corresponds to the current experimental state of the art in which the same device functions both as a processor and a network device. The second corresponds to a future architecture that separates these two functions over two distinct devices. We model these architectures as continuous-time Markov chains and compare their quality of executing quantum operations and producing entangled quantum states as functions of their memory lifetimes, as well as the time that it takes to perform various operations within each architecture. As an illustrative example, we apply our analysis to architectures based on Nitrogen-Vacancy centers in diamond, where we find that for present-day device parameters one architecture is more suited to computation-heavy applications, and the other for network-heavy ones. We validate our analysis with the quantum network simulator NetSquid. Besides the detailed study of these architectures, a novel contribution of our work are several formulas that connect an understanding of waiting time distributions to the decay of quantum quality over time for the most common noise models employed in quantum technologies. This provides a valuable new tool for performance evaluation experts, and its applications extend beyond the two architectures studied in this work.

1. Introduction

Quantum communication promises to fundamentally enhance internet technology by enabling application capabilities that are impossible to attain classically. On the one hand, quantum communication could be used to link quantum processors at large distances, enabling quantum internet [1,2] applications such as secure communication [3,4], improved clock synchronization [5], or secure delegated quantum computation in the cloud [6]. On the other hand, quantum communication could connect quantum processors at short distances in order to link several smaller quantum processors together to form one more powerful quantum computing cluster [7].

* Corresponding author.
E-mail addresses: g.s.vardoyan@tudelft.nl (G. Vardoyan), m.d.skrzypczyk@tudelft.nl (M. Skrzypczyk), s.d.c.wehner@tudelft.nl (S. Wehner).

https://doi.org/10.1016/j.peva.2021.102242

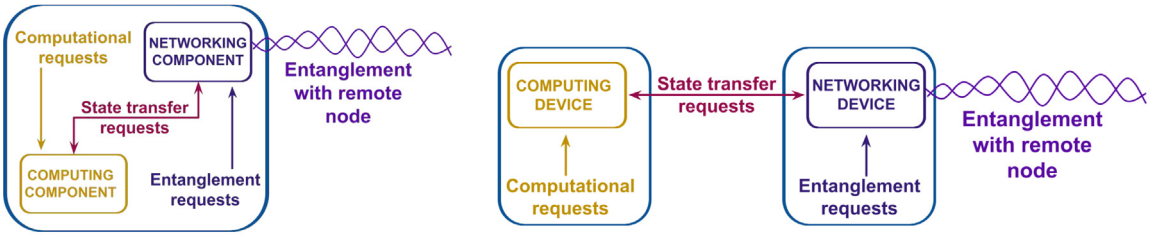
To support distributed quantum applications, the architecture of a quantum network node should be capable of two key functions: first, it should enable local quantum computation, *i.e.*, the execution of quantum gates and measurements, at each end node [2] in the network on which applications are run. Second, it should enable the generation of quantum entanglement between any two nodes in such a network. Entanglement is a special property of the state of two quantum bits (qubits), which cannot be simulated using any form of classical communication between the nodes. A typical quantum network application consists of both local quantum computations and the generation of entanglement, where different applications may have more demand for local quantum processing, or for entanglement generation. An example of an application that is computation-heavy is secure delegated quantum computation [6]. In contrast, quantum key distribution (QKD) [3,4,8] forms an example of an application that is network-heavy, *i.e.*, it is dominated by entanglement generation and the only local operations are measurements. Balancing local and networked operations (entanglement generation) can also be important in the efforts to build a quantum repeater [9–12], *i.e.*, a special quantum node that can eventually enable entanglement generation over arbitrarily long distances [13,14]. In this case, proposals for such repeaters employ both local quantum operations (*e.g.*, to perform entanglement purification [15–17]), as well as entanglement generation with neighboring repeater nodes.

When analyzing the performance of quantum networks, one is typically interested in understanding traditional performance metrics, such as the throughput or latency of entanglement generation and local gate execution. Importantly, however, the performance analysis of quantum technologies also demands a characterization of the *quality* of the quantum execution, *i.e.*, how noisy quantum states and operations are. Such a characterization is motivated both by long-term fundamental aspects of quantum applications, as well as the more short-term technological limitations of present-day quantum devices. In the classical world, a system is typically constructed in such a way that all errors are eliminated towards the application [18]. That is, an application sees essentially noise-free network transmissions and CPU operations. For many quantum network applications, however, noise-free transmission and quantum gate execution are not compulsory. A good example is QKD [3,4,8], where noise at the quantum level is dealt with using classical error correction, after measuring the quantum state, in a way that is specific to the application.

In quantum networked systems, fundamental tradeoffs exist between the quality of the quantum execution, and standard performance metrics such as throughput and latency. A key performance metric in a quantum network is the quality of entanglement (see Section 2.2.4) being generated between two remote network nodes, where one can choose to trade a higher throughput of entanglement generation, against a lower quality of the resulting entanglement and vice versa [18]. On a quantum processor, we furthermore want to understand the quality of a quantum gate's execution, and consequently the quality of the quantum program being executed. If quantum devices were perfect, a quantum gate could be executed with perfect quality, that is, the output is precisely as intended and no noise has occurred. In practice, however, technological limitations mean that gates on all present-day quantum processing platforms are noisy. Such noise can stem from inherent imperfections of the device (constant noise), as well as a time-dependent contribution that depends on the waiting time before the quantum state can undergo further processing. The latter form of noise is especially relevant when analyzing networked quantum processors, as is the focus of this work, where we frequently need to wait for a signal from the remote node before processing can continue. However, it also arises when trying to analyze any form of scheduling algorithm on an advanced quantum processor. In the quantum literature, the quality of a quantum state is measured by its *fidelity*, and the quality of executing a gate by its *gate fidelity* (see Section 2.2.4). Intuitively, the fidelity is a number in the interval $[0, 1]$ that measures the closeness of the state (or gate) to a desired target implementation. The larger the fidelity, the closer we are to the target implementation, *i.e.*, the higher the quality of the quantum state (or gate). In this work, we focus on these *quantum performance measures* — specifically, we study gate fidelity in distributed quantum architectures, as well as the fidelity of entanglement generated by applications that run on them.

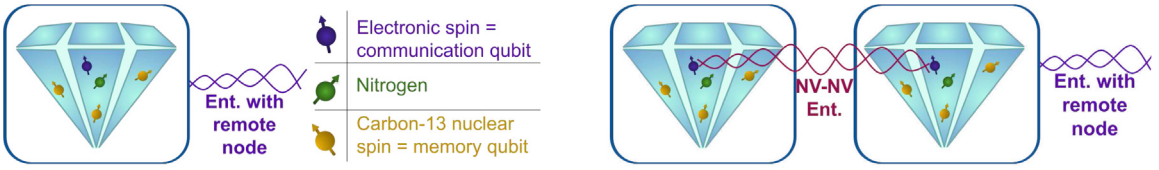
Given the need to perform both local quantum operations as well as network operations in order to realize distributed quantum applications, we here consider two different general architectures for interfacing a networked quantum processor to a quantum network (Fig. 1). In the first, which we call the single-device (SD) architecture, the same device is used to perform both network operations as well as local quantum computation (Fig. 1(a)). This is the case in all present-day implementations, such as networked quantum processors based on Nitrogen-Vacancy (NV) centers in diamond [19], or Ion Traps [20]. Abstractly, one can think of these as quantum processors that have two different types of qubits: communication qubits (networking component) with an optical interface for remote entanglement generation, and storage qubits which can only be used for local processing. Limits on experimental control typically prohibit the simultaneous execution of local (two-)qubit gates, and entangling operations. That is, while entanglement generation is in progress, local quantum processing is on hold, and vice versa. The time necessary for local gate execution only depends on the local processing speed. However, the time required for entanglement generation depends on the physical distance to the remote network node. Consequently, in a situation in which the remote node is at a distance, local processing may need to be suspended for a significant amount of time while entanglement generation is in progress.

In the second architecture, we hence consider a scenario in which the system is enhanced by the introduction of a dedicated network device solely used for the purpose of entanglement generation with remote network nodes (Fig. 1(b)); we refer to this as the double-device (DD) architecture. In this architecture, the network device is linked internally to the processor. While it is not important how this is achieved physically for our general analysis, we provide an example architecture in which both devices are based on NV centers in diamond (Fig. 2) where the internal interface



(a) single-device architecture (b) double-device architecture

Fig. 1. Two possible architectures for a quantum processor interfaced to a quantum network: in the first, the processor and the network device are the same device (*single-device (SD) architecture*). This device may have an internal logical or physical division into a computing or networking component. An example of a physical division is the use of a subset of its qubits for networking and others purely for computing. An example of a logical division is a scheduler switching between both functions but networking and computation are performed using the same qubits. In the second, two separate devices are used (*double-device (DD) architecture*). An application interacts with the system by making three types of requests: local quantum computations (on the computing component/device), network operations (entanglement generation), and movement (state transfer) of generated entanglement into the processor for further processing. The latter requires cooperation from both processing and network devices. For SD, a move could be achieved simply by transferring the state to another set of qubits on the same device. For DD, a move is much more complex, and can be realized *e.g.*, using entanglement generation between the processor and the network devices, followed by teleportation.



(a) the single-NV architecture (b) the double-NV architecture

Fig. 2. An example of SD and DD architectures based on Nitrogen-Vacancy centers in diamond. Here, the electron spin (purple) acts as an optical interface that can be used as a networking or computing component. Nitrogen (green) and the Carbon-13 spins (yellow) in the surrounding diamond material can be used as computing components. See Appendix D for further details about this platform.

is realized by teleporting the entanglement of the network device into the processor. This requires an additional step of producing entanglement between the network device and the processor to perform the teleportation transfer. Yet, since this entanglement needs to be produced only at very short (on-chip) distances, generation is fast. This means that remote entanglement generation via the external networking device and computations on the processor only need to be suspended for a short amount of time when the entanglement is transferred from the former to the latter. We remark that our analysis is fully general and could also be used to understand physical systems that divide the processor into networking and computing "zones" like segmented ion traps [21,22] or ones that use two different physical systems, such as for example NV centers in diamond for the processor, but a simpler device such as a quantum memory based on atomic ensembles [23] as the network device.

When deliberating such architectural choices, several considerations are of concern: first, it is clear that performance may depend on whether we execute a computation-heavy, or a network-heavy application. Indeed, it is clear that in the case of quantum key distribution, where there are no local quantum gates being executed and we simply measure the entanglement right away, the DD architecture may only introduce an unnecessary overhead in implementation. Second, we expect that the performance of both architectures depends on the inherent quality of the quantum devices used to realize them. One key concern is the ability of the quantum device to store quantum states during waiting times: a lower memory lifetime means that waiting times have a much larger impact on the quality of execution. Similarly, the quality of the interface between the processor and the network device is of concern in DD architectures, as it may reduce the quality of the entanglement being transferred. Finally, while the DD architecture may be of great intuitive appeal, it is much more cumbersome to realize experimentally since one additional device must be constructed. This raises a very practical question as to what achieves more benefit to application performance: implementing the double-device architecture, or investing efforts into improving the quality of the components (*e.g.*, to achieve higher memory lifetimes) in the single-device architecture.

Here, we make the following contributions in analyzing the two architectures:

- We provide mathematical formulas for computing the gate and entanglement fidelities for standard quantum noise models. These formulas can be applied to any quantum performance analysis problem, where one would like to understand how a waiting time t affects the quality of quantum gates and entanglement. As such, they allow standard methods from performance analysis that determine the waiting time distribution to be carried over to the quantum domain.

- For the two architectures introduced above, we determine the most defining characteristics and operational features. We then incorporate these features into a model that is representative of both architectural designs — specifically, we employ a continuous-time Markov chain (CTMC) to model entanglement generation in a regime where local quantum computation consumes negligible time. This is well motivated in the regime where the distance between processors is large as in a quantum internet, and the time required to produce entanglement dominates with respect to the time to perform local quantum gates. In this case, we obtain analytical expressions for the qubit waiting time distribution, subsequently allowing us to apply our fidelity computation method to obtain expressions for the average gate and entanglement fidelities for the two architectures in closed form. We later relax the assumption that local gates take negligible time, and explore the effects of more time-consuming computation via simulation. The latter is relevant when the processors are physically close.
- Using the aforementioned analytical results, we determine the strengths and weaknesses of the two architectures. Our analysis can be used to examine general tradeoffs between the quality of the quantum devices, the application behavior (computation or network-heavy), and the resulting fidelities for quantum gates and entangled states being produced. In a regime where DD quantum state transfer operations are more noisy than SD ones (*e.g.*, when they rely on imperfect gates), we find that the SD architecture is the more suitable option for applications that are network-heavy, while the DD architecture benefits computation-heavy applications. While the DD architecture outperforms the SD design in terms of average gate quality, its more complex design makes it harder to implement in practice. We provide sufficient conditions indicating how much the quantum memory lifetime of the components used to implement the SD architecture would need to be improved, in order to achieve the same performance as the DD architecture.
- We apply our analytical techniques to evaluate the performance of the two architectures under the assumption that they are realized using the Nitrogen-Vacancy center in diamond platform — a strong candidate for implementing near-term and future networked quantum nodes [24–27]. We explore the effect of state transfer operations on entanglement fidelity and where possible, present the pre-move and post-move entanglement fidelities in closed form. We validate our analytical results with NetSquid [28], a discrete-event quantum network simulator.

The rest of this paper is organized as follows: in Section 2, we discuss related work and cover the relevant quantum background. In Section 3, we introduce the CTMC that is used to model both architectures, and discuss the modeling assumptions. In Section 4 we determine the amount of time that a qubit must spend waiting in storage before it is processed. This waiting time distribution, along with a noise model, can be used to obtain the average gate and entanglement fidelities — in Section 5, we introduce a method for accomplishing this in a general setting and subsequently apply it to the architectures in Fig. 2. In Section 6 we show that when the two architectures have memories of identical quality, the DD architecture always outperforms the SD architecture in terms of average gate fidelity. Interestingly, it is possible that an SD-architecture device with better memories (and thus with a similar performance to a DD device with poorer quality memories) may be the more economical option in terms of manufacturing cost. For this reason, in Section 6 we also present sufficient conditions that, if satisfied, ensure the SD outperforms the DD architecture in terms of gate fidelity. In Section 7, we present analytical and simulation results and make numerical observations in a variety of settings. We make concluding remarks and discuss extensions of the problem in Section 8.

2. Background and related work

2.1. Related work

In practice, for any physical platform implementing a networked quantum processor, the quality of quantum gates and states is determined experimentally (see *e.g.*, [25–27,29–31], among a multitude of others). The objective of these measurements was to characterize one specific setup, but not to explore tradeoffs of potential architectural designs. For a string of quantum repeaters with the goal of producing entanglement over long distances, some analytical studies exist that characterize the quality of very specific quantum states, and study their distribution to guarantee a minimum threshold quality see, *e.g.*, [32–35]. Some analytical studies also exist for the so-called quantum switch, [36–38], wherein the authors study the maximum possible rate of entanglement switching and the expected number of entangled qubits in storage. These works are very different in spirit since they focus on the creation of quantum entanglement over long distances, and not on tradeoffs between network and computation operations as we do here. We emphasize that in this work we do not assume that the quantum architecture is used for any specific purpose, and make very few assumptions on the physical platforms used to realize potential architectures. Instead, our goal is to abstract these details in the form of configurable modeling parameters, *e.g.*, the demand for entanglement, or the rate at which it is successfully generated. At the time of writing, we are also not aware of a study that considers the interactions between quantum computation and networking within a single system, and how contention for resources and processing time affects fidelity.

2.2. Quantum background

2.2.1. Qubits, quantum states, and quantum gates

Here, we provide the necessary formalism needed in this work, and refer to *e.g.*, [39] for a more in-depth introduction. Quantum information is encoded using quantum bits, or *qubits*, in contrast to the usage of bits in traditional computing.

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$$

$$Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

Fig. 3. The Pauli operators.

In addition to holding information in the form of discrete values such as 0 or 1, qubits may hold *quantum states* that are linear combinations of these values. A (pure) quantum state can be expressed as a vector $|\psi\rangle \in \mathbb{C}^d$ of length 1, where d is often considered to be finite dimensional in quantum technologies. For n qubits the dimension is $d = 2^n$, where it is customary to label basis elements of $\mathbb{C}^d$ by strings $x = x_1, \ldots, x_n \in \{0, 1\}^n$. The Dirac notation $|\cdot\rangle$ is used to represent a vector and is referred to as a *ket* while the conjugate transpose, $\langle\cdot| = |\cdot\rangle^\dagger = (|\cdot\rangle^*)^T$, is referred to as a *bra*.

Quantum information is manipulated through the application of quantum gates. A quantum gate G is represented by a matrix $G \in \mathbb{C}^{d\times d}$, where G is unitary, *i.e.*, $GG^\dagger = G^\dagger G = \mathbb{I}$, where $\mathbb{I}$ is an identity matrix of dimension d. Applying a quantum gate G gives us the state $|\psi'\rangle = G|\psi\rangle$. Common quantum gates include the single-qubit Pauli operators (Fig. 3). Given the states of n individual qubits $|\psi_1\rangle, \ldots, |\psi_n\rangle$ the joint state of all n qubits is given by the tensor (Kronecker) product of the individual vectors $|\psi\rangle = |\psi_1\rangle \otimes \cdots \otimes |\psi_n\rangle$. Similarly, applying gates $G_1, \ldots, G_n$ to n individual qubits results in the application of the overall operation $G = G_1 \otimes \cdots \otimes G_n$.

Quantum computing and networking applications are realized by applying a series of quantum gates to one or several qubits and then performing a *measurement* of the qubits to read out information in the quantum states.

2.2.2. Noisy quantum states and operations

Noisy quantum states. A convenient way of representing a quantum state $|\psi\rangle$ is as a *density matrix* $\rho = |\psi\rangle\langle\psi|$ which is obtained by taking the outer product of the ket and the bra of the state. Importantly, the density matrix formalism allows for the expression of noisy quantum states. For example, a probabilistic process that prepares a desired state $|0\rangle\langle 0|$ with probability $1 - p$, but fails and instead prepares $|1\rangle\langle 1|$ with probability p, results in a noisy quantum state $\rho = (1-p)|0\rangle\langle 0| + p|1\rangle\langle 1|$. In general, the set of all quantum states on a d-dimensional quantum system (including noisy ones) corresponds to the set of matrices

$$\mathcal{S} = \left\{\rho \in \mathbb{C}^{d\times d}, \rho \succ 0 \text{ is positive semi-definite and normalized } \mathrm{Tr}(\rho) = 1\right\}. \tag{1}$$

Noise processes. Using this formalism we can now express the effect of noise on a quantum state. As an example, imagine a noise process that transforms a quantum state ρ_{initial} that is placed into a quantum memory at time $t = 0$, into a noisy quantum state ρ_{noisy} after a waiting time t. Qubits are susceptible to environmental noise that can inadvertently change their quantum state. Such noise can arise due to imperfect shielding of qubits from external influence as well as imperfect implementations of quantum gates. Mathematically, the set of all possible noise processes corresponds to the set of completely positive trace preserving maps (CPTPM) $\Lambda : \mathcal{S} \to \mathcal{S}$. The effect of environmental interaction on quantum states over time is often referred to as *decoherence* and is modeled through the use of *noise models* describing Λ. Common noise models include $\Lambda = \mathcal{D}$ depolarizing noise,

$$\mathcal{D}_t(\rho) = (1 - 3p)\rho + pX\rho X + pY\rho Y + pZ\rho Z = (1 - 4p)\rho + 4p\frac{\mathbb{I}}{2}, \tag{2}$$

which drives a quantum state towards the maximally noisy state, also called the maximally mixed state $\frac{\mathbb{I}}{2}$. This state is the quantum equivalent of white noise. Here, time dependence is often expressed by letting $p = \frac{1}{4}(1 - e^{-\frac{t}{T}})$, for a fixed T characterizing the quantum memory storing the quantum state. This allows one to express the noise incurred in a quantum memory storing a qubit, after a waiting time of t has elapsed. A model of depolarizing noise is often used as a worst case estimate, when the physical noise process is insufficiently characterized.

In the literature describing implementations of quantum memories, we often have more information about the noise process of the quantum memory device. This noise is generally modeled as dephasing and damping noise (or a combination of both). Dephasing noise is expressed as

$$\mathcal{P}_t(\rho) = (1 - p)\rho + pZ\rho Z, \tag{3}$$

where similarly $p = \frac{1}{2}(1 - e^{-\frac{t}{T_2}})$ is used to express time-dependence, for a fixed T_2 characterizing the memory. This can be understood as an analogue of the classical binary symmetric channel, where a flip operation (here, Z) is applied with some probability p.

Another common model is the amplitude damping noise channel

$$\mathcal{A}_t(\rho) = M_0\rho M_0^\dagger + M_1\rho M_1^\dagger, \tag{4}$$

where M_0, M_1 have the form

$$M_0 = \begin{bmatrix} 1 & 0 \\ 0 & \sqrt{1-\gamma} \end{bmatrix}, \qquad M_1 = \begin{bmatrix} 0 & \sqrt{\gamma} \\ 0 & 0 \end{bmatrix}, \tag{5}$$

and $\gamma = (1 - e^{-\frac{t}{T_1}})$ for a fixed T_1 characterizing the effects of the amplitude damping channel. This can be understood as the quantum analogue of a noisy channel of one-sided error which preserves $|0\rangle\langle 0|$, but damps $|1\rangle\langle 1|$ to $|0\rangle\langle 0|$ with an error probability γ.

In most physical implementations of quantum devices, both $\mathcal{P}_t$ and $\mathcal{A}_t$ occur and the noise is described by a composite model

$$\mathcal{C}_t(\rho) = \mathcal{P}_t(\mathcal{A}_t(\rho)) = (1-p)\mathcal{A}_t(\rho) + pZ\mathcal{A}_t(\rho)Z \tag{6}$$

$$= (1-p)\left(M_0\rho M_0^\dagger + M_1\rho M_1^\dagger\right) + pZ\left(M_0\rho M_0^\dagger + M_1\rho M_1^\dagger\right)Z \tag{7}$$

where, in general, $T_2 < T_1$ [25,39–42]. Larger values of T, T_1, and T_2 correspond to a quantum memory with a longer memory lifetime.

Noisy quantum gates. The effect of a noisy quantum gate can be described in an entirely analogous fashion. Note that in terms of the density formalism, the effect of applying a gate G on a quantum state can be expressed as

$$G(\rho) = G\rho G^\dagger \ , \tag{8}$$

where we follow convention and use G both to denote the unitary matrix, as well as the CPTPM as indicated by context. When modeling noise in quantum gates it is customary to model a noisy implementation $\mathcal{E}$ as the ideal gate, followed by possibly time dependent noise $\mathcal{N}_t$. That is, $\mathcal{E} = \mathcal{N}_t \circ G$, where G is the ideal implementation of the gate. We will follow this custom here.

As an example, consider a situation in which we perform the gate $G = X$, but then incur a waiting time of t before the next quantum operation is applied. If the total noise process (inherent noise in the gate, plus noise due to waiting) is described by dephasing and damping noise $\mathcal{C}_t$, then the initial state ρ_{initial} is transformed to the noisy state

$$\rho_{\text{noisy}} = \mathcal{C}_t\left(X\ \rho_{\text{initial}}\ X^\dagger\right) \tag{9}$$

after the waiting time of t has elapsed.

2.2.3. Entanglement

Most quantum applications rely on a special property known as *entanglement* that qubits can share. Mathematically, a state $\rho_{AB} \in \mathbb{C}^{d_A \times d_A} \otimes \mathbb{C}^{d_B \times d_B}$ of a combined quantum system of nodes (or qubits) A and B is called separable if and only if it can be written as a classical mixture (*i.e.*, convex combination) of tensor products of single-node states (*i.e.*, $\rho_{AB} = \sum_j p_j \sigma_A^j \otimes \tau_B^j$ for some distribution $\{p_j\}_j$, and states $\{\sigma_A^j\}_j$ on A and $\{\tau_B^j\}_j$ on B). Intuitively, separable states have only classical correlations between A and B, since we may toss a coin according to p_j and then prepare individual states on A and B without any form of quantum interaction between them. Any state ρ_{AB} that is not separable is called *entangled*. In general, $|\Psi\rangle = \frac{1}{\sqrt{d}}\sum_j |j\rangle_A \otimes |j\rangle_B$ is a maximally entangled state between two d-dimensional systems A and B. Such entangled states form the primary building block of most quantum network applications.

2.2.4. Quantum quality measure: Fidelity

Fidelity of a state. The fidelity of a quantum state ρ measures how well this state approximates a specific target state $|\psi\rangle$. It is the relevant quantity used to understand how a fixed noise process (*e.g.*, during the preparation of the state) affects its quality, or how the quality of an already prepared state decreases as a function of a waiting time t that this state spends in a quantum memory. Specifically, the fidelity F of a state ρ to a target state $|\psi\rangle$ is defined as [43,44]

$$F = F(\rho, |\psi\rangle) = \langle\psi|\rho|\psi\rangle \tag{10}$$

such that $F = 1$ iff ρ is identical to the target state $|\psi\rangle$. The fidelity F lies in the interval [0, 1] and larger values of F indicate that ρ is closer to the target state $|\psi\rangle$.

Gate fidelity. The average gate fidelity measures how well a real-world implementation $\mathcal{E}$ approximates a desired target gate G, and is defined as (see *e.g.*, [45])

$$F_{\text{orig}}(\mathcal{E}, G) = \int d\Psi \langle\Psi|G^\dagger\mathcal{E}(|\Psi\rangle\langle\Psi|)G|\Psi\rangle \ , \tag{11}$$

where $d\Psi$ is the Haar (uniform) measure on the set of quantum states. That is, the gate fidelity measures how well the implementation approximates the target gate when applied to a specific input state $|\Psi\rangle$, averaged over all possible input states.

When $\mathcal{E} = \mathcal{N}_t \circ G$ (see above) for some time-dependent noise $\mathcal{N}_t$, we will also use the shorthand

$$F(\mathcal{N}_t, G) = F_{\text{orig}}(\mathcal{N}_t \circ G, G) \tag{12}$$

Table 1
Variables used in the quantum architecture model and their descriptions.

Variable	Description
λ_e	Entanglement request arrival rate
μ_e	Entanglement generation rate
λ_m	State transfer request arrival rate
μ_m	State transfer completion rate
λ_c	Computation request arrival rate
μ_c	Computation completion rate

to denote the resulting average fidelity. Eq. (12) is the relevant quantity when we are interested in the question: provided we had to wait time t after executing the gate G (*e.g.*, due to a scheduling decision), what is the resulting gate fidelity? We remark that since G is unitary, the case where time-dependent noise is applied *before* the execution of the gate G instead reduces to simply studying $\int d\Psi \langle\Psi|\mathcal{N}_t(|\Psi\rangle\langle\Psi|)|\Psi\rangle$. As we will see, our later formulas apply to both cases.

Entanglement fidelity. The entanglement fidelity [46] measures the quality of an initially maximally entangled state after it was stored in a noisy memory on node B (or A) for time t. Specifically, for a noise process $\mathbb{I}_A \otimes \mathcal{N}_t$ (no noise on A, and time-dependent noise $\mathcal{N}_t$ on system B), the entanglement fidelity is defined as

$$F_e(\mathcal{N}_t) = \langle\Psi|\mathbb{I}_A \otimes \mathcal{N}_t\left(|\Psi\rangle\langle\Psi|\right)|\Psi\rangle \ , \tag{13}$$

where $|\Psi\rangle$ is the maximally entangled state defined above. We remark that the case of noise on A and B can always be dealt with by observing that for any matrix M applied to A, this can be translated to applying M^T to B: $M_A \otimes \mathbb{I}_B|\Psi\rangle = \mathbb{I}_A \otimes M_B^T|\Psi\rangle$. That is, noise on A and B can be understood by applying both types of noise to the system B in succession. It turns out that the gate fidelity, and entanglement fidelity are related as [46]

$$F_{\text{orig}}(\mathcal{E}, G) = \frac{dF_e(\mathcal{E}) + 1}{d+1} \ , \tag{14}$$

where d is the dimension of A and B. For qubits, $d = 2$.

3. Modeling the architectures

We first provide a summary of the architecture attributes to be considered in the modeling and analysis of our problem. First, motivated by the limits of implemented quantum devices, in the SD architecture all operations must be performed sequentially: *e.g.*, computation may not be performed when entanglement generation or a state transfer are in progress. In the DD architecture, entanglement generation and computation (assumed to be independent of each other) may be performed in parallel, as these operations take place in separate devices. When a state transfer operation is in progress, however, both devices in the DD architecture must wait until its completion before servicing another computation or entanglement generation request. This is motivated by the same limit that prevented the simultaneous execution of network and computation operations in the SD architecture: a network operation is needed at the computation device in the DD architecture to transfer the entangled state, but this time only for the time needed to produce entanglement with the very close network device. In both architectures, a state transfer is required before a new entanglement request may be serviced, as this frees up the communication qubit required for further entanglement generation attempts.

To define the state space of our problem, it is helpful to classify the processes that make use of the quantum processors as follows: (*i*) entanglement generation, (*ii*) state transfer operations (we interchangeably refer to these as moving operations), and (*iii*) computation. Each class of operations (or jobs) is associated with an arrival rate and a processing rate, as specified in Table 1. We assume that all request arrivals are Poisson and all processing times are exponentially-distributed. λ_e represents the demand for entanglement, *i.e.*, the entanglement request rate either from a user or an application. The corresponding parameter μ_e represents the rate at which (remote) entanglement is generated — this is a function of the link length. Entanglement generation attempts at the elementary link level are often modeled as Bernoulli trials with some fixed success probability p_{gen} for each attempt — see, *e.g.*, [32–34,37]. In [36], the authors model the time between successful generation attempts as an exponentially-distributed random variable (r.v.) to accommodate their use of a CTMC when modeling a quantum network node; we adopt their convention here.

When entanglement is successfully generated, we assume that the state of the entangled qubit is eventually moved from the networking component to the computing component for processing, *e.g.*, in the single-NV example, the state is transferred from the electronic spin to the carbon spin. Since this moving operation may not be requested immediately, we introduce λ_m as the moving request rate. The time to physically perform such moving operations is exponentially-distributed with parameter μ_m; in general μ_m is lower for DD architectures than for SD ones, as the former requires more complex gate sequences to perform device-device state transfers: *e.g.*, in the double-NV example, NV-NV entanglement generation is required. Finally, computational jobs arrive according to a Poisson process with parameter λ_c, and their processing times are exponentially-distributed with parameter μ_c. Note that, save for entanglement generation, our use

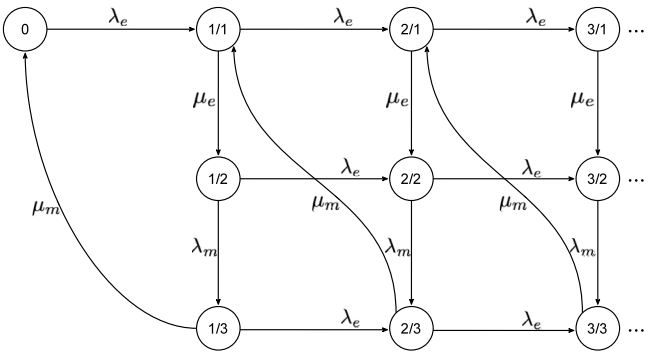


Fig. 4. A CTMC to model the SD and DD architectures.

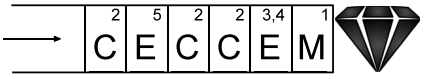


Fig. 5. Example queue occupancy: C, E, and M represent computational, entanglement, and moving job, respectively. The numbers in each slot represent the processing order: job M is currently being executed, but all C jobs may be processed before the next E request's processing begins. The second E request is fifth to be processed, since the first E request must be followed by a moving operation.

of the exponential distribution in modeling processing times is largely motivated by the resulting simplicity of the model and its analysis. A more realistic way to model state transfers for an SD design implemented with an NV center in diamond, for instance, would be to assume that their service times are deterministic (albeit, for the double-NV design, the use of the exponential distribution is well-justified due to the need to generate on-chip entanglement when servicing state transfer requests). In practice, the sojourn time distribution in each state is determined not only by the application, but also by the physical platform (*e.g.*, NV center in diamond, ion traps, atomic ensembles). Depending on the latter, it is possible that even the time it takes to perform a local state transfer is a random variable. In the interest of keeping the assumptions as general as possible and the results interpretable, we opt for the exponential distribution. However, if necessary, one may accommodate arbitrary service time distributions by modeling the architecture as a semi-Markov process. Depending on the specific application, it may also be appropriate to include computational jobs as an additional phase of the QBD process. Several other extensions of our model may also be considered depending on the use case – see Section 8 for further discussion – but lie outside the scope of this work.

In summary, the processing rates μ_e, μ_m, and μ_c depend on the properties of the architecture, while the request arrival rates λ_e, λ_m, and λ_c depend on application demands. When the λ_e value is high or μ_e is low, the application may be thought of as networking(entanglement)-heavy, while high values of λ_c and low values of μ_c correspond to a computation-heavy application. In general, μ_c is much greater than μ_e and even μ_m for the DD architecture, as local gates are far less time-consuming than entanglement generation in a network where processors are separated by large distances. For this reason, when constructing the model we make a simplifying assumption that $\mu_c = \infty$, *i.e.*, we assume that computational job processing times are negligible. As a consequence, computational jobs may be processed whenever the processor(s) would otherwise be idle (*i.e.*, waiting for a moving request to arrive), as well as in-between events (*e.g.*, immediately after the completion of a moving request but right before the entanglement generation of the next entanglement job) as their processing does not affect the rest of the system. As a result, a computational job need only to wait for the completion of a single event – either entanglement generation or a moving request – and as soon as this event has been completed, all computational jobs that are in the queue may be processed instantaneously.[1] An example is shown in Fig. 5. These modeling assumptions ultimately allow us to obtain all necessary performance measures in closed form; however, in Section 7 we remove the assumption on negligible processing times for computational jobs and observe the effects on the average fidelity numerically. Lower values of μ_c may be used to model "atomic" gate sequences which must not be interrupted by any other task or operation.

With the aforementioned assumptions, we may model both the SD and DD architectures as $M/HYPO_3/1$ queueing systems, where the arrivals correspond to entanglement requests and are a Poisson process with rate λ_e, and the service times are hypo-exponentially distributed with three service stages each of which are exponentially-distributed with parameters μ_e, λ_m, and μ_m. Fig. 4 depicts the CTMC representing this queueing system: each state of the form N/k corresponds to N outstanding entanglement requests in the system, with the first job in the kth stage of its processing. Entanglement requests are processed according to a first-in, first-out (FIFO) policy: the first stage ($k = 1$) is entanglement generation, the second ($k = 2$) is awaiting the arrival of a moving request, and the third ($k = 3$) is the execution of a

[1] For some physical platforms, additional simplifying assumptions on computational jobs may be necessary — see Appendix D for a discussion.

moving request. Note that the next entanglement request cannot begin processing until all three stages of the previous request have been completed, since the communication qubit must be freed before entanglement generation may be attempted again. State 0 corresponds to the case with no outstanding entanglement requests.

Note from the CTMC that a computational request only waits whenever its arrival coincides with the system being in states of the form $N/1$ and $N/3$ in the SD architecture, or if the system is in a state $N/3$ in the DD architecture. Thus, by the memoryless property of the exponential distribution, a computational job's waiting time distribution is given either by $f_e(t) = \mu_e e^{-\mu_e t}$ or by $f_m(t) = \mu_m e^{-\mu_m t}$, depending on the state of the system upon arrival. We may obtain the probabilities of arrival into states $N/1$ and $N/3$ from the stationary distribution of the CTMC. We remark that when computing the entanglement fidelity, we are interested in the amount of time a newly-entangled qubit must wait for a state transfer request to arrive, *i.e.*, we require the *waiting* time distribution, and not the sojourn time distribution of that qubit (which also includes the amount of time it takes to perform the moving operation). The reason is that the specific gate sequence used to perform the state transfer in a given architecture already implicitly accounts for the time it takes to execute the gates.

A note on mathematical notation: in the remainder of the paper, we use superscripts $^{(1)}$ and $^{(2)}$ to denote parameters corresponding to the SD and DD architectures, respectively. *E.g.*, $\mu_m^{(1)}$ corresponds to the moving rate in the SD architecture, while $T_1^{(2)}$ and $T_2^{(2)}$ refer to the memory lifetimes of the DD architecture.

4. Waiting time distributions

The CTMC shown in Fig. 4 has been studied in literature: specifically, it is a quasi birth–death (QBD) process with the special property that one of the blocks in its generator matrix is a rank 1 matrix, allowing us to compute the rate matrix explicitly using the results in [47]. For completeness, we include the rate matrix derivation and ergodicity condition for this Markov chain in Appendix A.1. For the following computations, assume that the mean drift condition

$$\frac{1}{\lambda_e} > \frac{1}{\mu_e} + \frac{1}{\lambda_m} + \frac{1}{\mu_m} \tag{15}$$

is satisfied so that a stationary distribution exists. Recall from the discussion in Section 3 that to determine the waiting time distribution of a computational job, it suffices to compute the stationary probabilities of states $N/1$ and $N/3$, which we label $\pi_{N/1}$ and $\pi_{N/3}$, respectively. Specifically, rather than having to derive the individual stationary probability of each state in phase 1 or 3, we need only to compute the aggregate probabilities $\sum_{N=1}^{\infty}\pi_{N/1}$ and $\sum_{N=1}^{\infty}\pi_{N/3}$, since a computational request's waiting time has no dependence on N, but only on the phase (stage) in which the QBD process has been found upon arrival. In Appendix A.2, we show that $\sum_{N=1}^{\infty}\pi_{N/1} = \lambda_e/\mu_e$ and $\sum_{N=1}^{\infty}\pi_{N/3} = \lambda_e/\mu_m$.

We are now ready to compute the waiting time distributions of computational jobs in both types of architectures. Recall that in the SD architecture, a computational job may be processed immediately, as long as there is no ongoing entanglement generation or moving job in progress. Specifically, if a computational request arrives while the QBD process is in a state $N/2$, then it is processed immediately (the waiting time is zero), and otherwise, the request is queued behind the ongoing entanglement or moving request. Thus, the waiting time (conditioned on the stage k) for a computational job in the SD architecture is given by $W_1 = S_e\mathbb{1}_{\{k=1\}} + S_{m_1}\mathbb{1}_{\{k=3\}}$, where $\mathbb{1}$ is the indicator function, S_e is an exponentially-distributed r.v. with mean $1/\mu_e$ and probability density function (p.d. (f) $f_e(t)$, and S_{m_1} is an exponentially-distributed r.v. with mean $1/\mu_m^{(1)}$ and p.d.f. $f_{m_1}(t)$, with $\mu_m^{(1)}$ the rate of completing a moving request (after its arrival, meaning that this is the rate of solely executing the gates to fulfill a moving request) in the SD architecture. Then, by the law of total probability (*cf.* Eq. (2.26) in [48]), the marginal p.d.f. for the waiting time of computational jobs in the SD architecture is given by

$$f_{W_1}(t) = \sum_{N=1}^{\infty}\pi_{N/1}f_e(t) + \sum_{N=1}^{\infty}\pi_{N/3}f_{m_1}(t) + \left(1 - \sum_{N=1}^{\infty}\pi_{N/1} - \sum_{N=1}^{\infty}\pi_{N/3}\right)\delta(t) \tag{16}$$

$$= \lambda_e e^{-\mu_e t} + \lambda_e e^{-\mu_m^{(1)} t} + \left(1 - \frac{\lambda_e}{\mu_e} - \frac{\lambda_e}{\mu_m^{(1)}}\right)\delta(t), \tag{17}$$

where $\delta(t)$ is the Dirac delta function, defined as

$$\delta(t) = \begin{cases} +\infty, & t = 0, \\ 0, & t \neq 0, \end{cases} \quad \text{and satisfies} \quad \int_{-\infty}^{\infty}\delta(t)dt = 1.$$

Next, we consider the waiting time distribution of computational jobs in the DD architecture, wherein a computational request need only wait if a moving request is actively being executed. Letting f_{m_2} be the p.d.f. of an $\sim Exp(\mu_m^{(2)})$ r.v., this distribution is given by

$$f_{W_2}(t) = \sum_{N=1}^{\infty}\pi_{N/3}f_{m_2}(t) + \left(1 - \sum_{N=1}^{\infty}\pi_{N/3}\right)\delta(t) = \lambda_e e^{-\mu_m^{(2)} t} + \left(1 - \frac{\lambda_e}{\mu_m^{(2)}}\right)\delta(t). \tag{18}$$

When using f_{W_1} and f_{W_2} to compute average gate fidelity in Section 5.2, we will use the definition

$$\int_a^b f(x)\delta(x-c)dx = f(c), \tag{19}$$

where $a \leq c \leq b$ and $f(x)$ is a function continuous on the interval $[a, b]$.

Another quantity of interest is the waiting time distribution of a newly-entangled qubit while it awaits a state transfer request. For both architectures, this is given by $g(t) = \lambda_m e^{-\lambda_m t}$.

5. Formulas for computing quantum fidelities

We now provide several general formulas that can be used to link an understanding of waiting times t to the gate and entanglement fidelities for the standard noise models used to describe quantum devices. Our formulas for the gate fidelities can be applied to the situations described in Section 2.2.4, where we need to wait for a time t before or after executing a quantum gate. In this work, this waiting time occurs since we need to suspend quantum processing when performing network operations (see Section 3). However, we remark that our formulas are applicable to any other situations where such waiting times arise, such as for example in the analysis of algorithms for scheduling gates on a quantum processor. We emphasize that our formulas also apply to a situation where one would simply want to understand the average reduction in quality when storing a qubit in memory, which corresponds to applying the trivial gate $G = \mathbb{I}$.

Using the known link between gate and entanglement fidelities [46] in Eq. (14), these formulas can also be directly applied to understand how the quality of an entangled link decays as a function of a waiting time t. The use case in this work is to understand the decay of entanglement due to waiting for gate or move operations (see Section 3). However, it can also be applied to any other situation where a waiting time t is incurred before the entanglement can be processed.

Lemma 1. *Let $\mathcal{D}_t$, $\mathcal{P}_t$, $\mathcal{A}_t$ and $\mathcal{C}_t$ denote the depolarizing channel, dephasing channel, amplitude damping and composite channel for a single qubit respectively, as defined in Section 2.2.2. Let $t > 0$ and G be any quantum gate (unitary).*

- *For depolarizing noise $\mathcal{D}_t$ with $p = \frac{1}{4}\left(1 - e^{-t/T}\right)$*

$$F(\mathcal{D}_t, G) = \frac{1}{2}\left(1 + e^{-\frac{t}{T}}\right) . \tag{20}$$

- *For dephasing noise $\mathcal{P}_t$ with $p = \frac{1}{2}\left(1 - e^{-t/T_2}\right)$*

$$F(\mathcal{P}_t, G) = \frac{1}{3}\left(2 + e^{-\frac{t}{T_2}}\right) . \tag{21}$$

- *For amplitude damping noise $\mathcal{A}_t$ with $\gamma = (1 - e^{-\frac{t}{T_1}})$,*

$$F(\mathcal{A}_t, G) = \frac{1}{6}\left(3 + e^{-\frac{t}{T_1}} + 2e^{-\frac{t}{2T_1}}\right) . \tag{22}$$

- *For the composite noise model $\mathcal{C}_t$ using $\gamma = (1 - e^{-\frac{t}{T_1}})$ for $\mathcal{A}_t$ and $p = \frac{1}{2}\left(1 - e^{-t\left(\frac{1}{T_2} - \frac{1}{2T_1}\right)}\right)$ for $\mathcal{P}_t$ (to account for dephasing effects of $\mathcal{A}_t$ [28,39])*

$$F(\mathcal{C}_t, G) = \frac{1}{6}\left(3 + e^{-\frac{t}{T_1}} + 2e^{-\frac{t}{T_2}}\right), \tag{23}$$

where we assume $0 < T_2 \leq 2T_1$ so that Eqs. (21) and (22) are recovered as $T_1 \to \infty$ and $T_2 \to 2T_1$.

To the non-quantum expert, it may come as a surprise that the formulas above only depend on the noise, but not on the specific gate G. As we will see, this is simply a consequence of G being unitary, combined with the fact that we uniformly average over the set of all quantum states and this average does not change when a unitary is applied.

Given an understanding of the waiting time distribution, one may then readily compute the average gate fidelity due to waiting as

$$F_{avg}(\mathcal{N}_t, G) = \int dw(t) F(\mathcal{N}_t, G) , \tag{24}$$

where $dw(t)$ is the measure over waiting times t resulting from a specific model, and $\mathcal{N}_t$ is the quantum channel.

5.1. Derivation

We remark that there are several methods to obtain the same result, and for completeness we present a self-contained derivation using only elementary facts from quantum information theory in Appendix E.1. Here, we make use of a result

in [49] for the case where the quantum channels are applied to qubits, as relevant for the standard noise models above. From [49] we have that for any one qubit quantum channel $\mathcal{E}$ used to approximate a gate G that

$$F_{\text{orig}}(\mathcal{E}, G) = \frac{1}{2} + \frac{1}{12}\left(\text{Tr}\left[GXG^\dagger \mathcal{E}[X]\right] + \text{Tr}\left[GYG^\dagger \mathcal{E}[Y]\right] + \text{Tr}\left[GZG^\dagger \mathcal{E}[Z]\right]\right) \tag{25}$$

where X, Y and Z are the Pauli matrices defined in Section 2. When $\mathcal{E} = G \circ \mathcal{N}_t$ (*i.e.*, noise in the gate is modeled by applying first a noisy channel $\mathcal{N}_t$ followed by the ideal implementation of G) we have that since G is unitary ($G^\dagger G = GG^\dagger = \mathbb{I}$),

$$F_{\text{orig}}(G \circ \mathcal{N}_t, G) = \int d\psi \langle\psi| G^\dagger G \mathcal{N}_t(|\psi\rangle\langle\psi|) G^\dagger G |\psi\rangle = \int d\psi \langle\psi| \mathcal{N}_t(|\psi\rangle\langle\psi|)|\psi\rangle = F_{\text{orig}}(\mathcal{N}_t, \mathbb{I}) \,.$$

Using (25), the Pauli matrices, and the definitions of the quantum noise channels (see Section 2), matrix algebra then yields the claimed formulas above. For the case where $\mathcal{E} = \mathcal{N}_t \circ G$ (*i.e.*, the noisy gate is modeled by first applying the ideal gate G and then applying a noise process $\mathcal{N}_t$, a common convention in quantum technologies), we also obtain

$$F_{\text{orig}}(\mathcal{N}_t \circ G, G) = \int d\psi \langle\psi| G^\dagger \mathcal{N}_t(G|\psi\rangle\langle\psi|G^\dagger) G|\psi\rangle = \int d\psi \langle\psi| \mathcal{N}_t(|\psi\rangle\langle\psi|)|\psi\rangle = F_{\text{orig}}(\mathcal{N}_t, \mathbb{I}) \,,$$

where this time we have made use of the fact that the Haar (uniform) measure $d\psi$ on the set of quantum states is invariant under the application of a unitary G, since a unitary simply permutes the set of quantum states $G|\psi\rangle = |\psi'\rangle$. We remark that in quantum information operations do not generally commute and $\mathcal{N}_t \circ G \neq G \circ \mathcal{N}_t$ for most choices of G and $\mathcal{N}_t$.

5.2. Application to our problem

As discussed in Section 2.2, fidelity acts as a measure by which we may evaluate the performance of single- and double-device architectures. For computation requests, we determine an associated gate fidelity that reflects the quality of the quantum gate(s) that are applied in the computation. For moving requests, we consider the fidelity of the entanglement that is delivered to applications.

We evaluate the average gate fidelity (Eq. (24)) for computation requests on the SD and DD architectures using the composite noise model $\mathcal{C}$ and the waiting time distributions in (17), (18):

$$\begin{aligned} F^{(1)}_{avg}(\mathcal{C}, G) &= \int_0^\infty f_{W_1}(t) F(\mathcal{C}, G) dt \\ &= 1 - \frac{\lambda_e}{2}\left(\frac{1}{\mu_e} + \frac{1}{\mu_m^{(1)}}\right) + \frac{\lambda_e}{6}\left(\frac{2T_2}{\mu_e T_2 + 1} + \frac{T_1}{\mu_e T_1 + 1} + \frac{2T_2}{\mu_m^{(1)} T_2 + 1} + \frac{T_1}{\mu_m^{(1)} T_1 + 1}\right), \end{aligned} \tag{26}$$

$$F^{(2)}_{avg}(\mathcal{C}, G) = \int_0^\infty f_{W_2}(t) F(\mathcal{C}, G) dt = 1 - \frac{\lambda_e}{2\mu_m^{(2)}} + \frac{\lambda_e}{6}\left(\frac{2T_2}{\mu_m^{(2)} T_2 + 1} + \frac{T_1}{\mu_m^{(2)} T_1 + 1}\right), \tag{27}$$

where we use (19) to evaluate the integrals above. We also evaluate the average fidelity of the entanglement that is moved into memory. The waiting time distribution for moving requests for both architectures is given by $g(t) = \lambda_m e^{-\lambda_m t}$, so that

$$F_e(\mathcal{C}) = \frac{3F_{avg}(\mathcal{C}, I) - 1}{2} = \frac{1}{4}\int_0^\infty \lambda_m e^{-\lambda_m t}\left(3 + e^{-\frac{t}{T_1}} + 2e^{-\frac{t}{T_2}}\right) dt - \frac{1}{2} = \frac{1}{4} + \frac{\lambda_m}{4}\left(\frac{T_1}{\lambda_m T_1 + 1} + \frac{2T_2}{\lambda_m T_2 + 1}\right).$$

6. Analytical evaluation

In Section 4, we derived the waiting time distributions of computational jobs in each of the architectures, and in Section 5 we determined how these distributions translate into the average gate fidelity. Using these results, it is possible to compare the average gate fidelity of the SD to that of the DD architecture. Specifically, we will show that when computational job processing times are negligible (*i.e.*, when the queueing systems are both represented by the CTMC in Fig. 4) and the memories in both architectures are of identical manufacturing quality, then the DD architecture always outperforms the SD architecture in terms of the average gate fidelity. Our standard assumption that $\mu_e \leq \mu_m^{(2)}$ holds for the following discussion.

Proposition 1. *When $\mu_c = \infty$, the mean drift conditions (15) for the SD and DD architectures are satisfied, and $F(\mathcal{N}_t, G)$ is identical for both architectures, the DD architecture yields a higher average gate fidelity than the SD architecture.*

See Appendix B for a proof of Proposition 1.

It is worth emphasizing that the result in Prop. 1 holds when the two Markov chains modeling the architectures are stable; *i.e.*, while the DD architecture yields better performance for average gate fidelity, *cf.* 1 it is also more difficult to ensure its stability, since in general, $\mu_m^{(1)} > \mu_m^{(2)}$.

For the remainder of this section, we focus on the composite noise model for storage introduced in Section 5.2. Now, suppose that the characteristic memory times of the two architectures are not identical. As discussed previously, in such

cases it is possible that an SD architecture with higher-quality memories is the more cost-effective option while also yielding a higher average gate fidelity than a DD architecture with memories of poorer quality. This brings up a natural question: given a DD architecture with fixed characteristic memory times $T_1^{(2)}$ and $T_2^{(2)}$, what conditions must the memory lifetimes of an SD architecture satisfy in order to outperform the former in terms of average gate fidelity? From Eqs. (26) and (27), we see that $F_{avg}^{(1)} > F_{avg}^{(2)}$ when

$$\frac{1}{\mu_e}+\frac{1}{\mu_m^{(1)}}-\frac{1}{3}\left(\frac{2T_2^{(1)}}{\mu_e T_2^{(1)}+1}+\frac{T_1^{(1)}}{\mu_e T_1^{(1)}+1}+\frac{2T_2^{(1)}}{\mu_m^{(1)} T_2^{(1)}+1}+\frac{T_1^{(1)}}{\mu_m^{(1)} T_1^{(1)}+1}\right) < \frac{1}{\mu_m^{(2)}}-\frac{1}{3}\left(\frac{2T_2^{(2)}}{\mu_m^{(2)} T_2^{(2)}+1}+\frac{T_1^{(2)}}{\mu_m^{(2)} T_1^{(2)}+1}\right). \quad (28)$$

For the following discussion, it is useful to keep in mind that for a constant $c > 0$, $\lim_{x\to\infty} x/(cx+1) = 1/c$. This implies that both sides of the inequality above are positive. Recall from previous discussion that the remote entanglement generation rate μ_e is a smaller value than the moving rates $\mu_m^{(1)}$ and $\mu_m^{(2)}$. From (28), it is easy to see that the value of μ_e plays a significant role in determining how much the memory lifetimes must compensate to improve the performance of the SD architecture.

To obtain a more interpretable and intuitive understanding of how high the SD architecture memory times must be, we derive a sufficient condition in terms of solely $T_2^{(1)}$, $T_1^{(2)}$, μ_e, $\mu_m^{(1)}$, and $\mu_m^{(2)}$. This condition serves as a good bound to (28) when the memory lifetimes $T_1^{(2)}$ and $T_2^{(2)}$ are not too far apart from each other, and becomes tighter as μ_e increases.

Proposition 2. *Assume μ_e is greater than one and $\mu_e < \mu_m^{(2)} < \mu_m^{(1)}$. Then the SD architecture on average achieves a higher gate fidelity than the DD architecture when*

$$T_2^{(1)} > \frac{\mu_m^{(2)}(\mu_m^{(2)}T_1^{(2)}+1)(\mu_e+\mu_m^{(1)})}{\left(\mu_e\mu_m^{(1)}\right)^2}-\frac{\mu_e+\mu_m^{(1)}}{\sqrt{2}\mu_e\mu_m^{(1)}}. \quad (29)$$

See Appendix C for a proof of this proposition.

Condition (29) tells us that the faster the entanglement generation rate, the smaller the memory requirements are on the SD architecture to ensure that it outperforms the DD architecture. This is intuitive since in the SD architecture, entanglement generation is the most time-consuming and therefore the most detrimental operation to the gate fidelity. Condition (29) also tells us that the faster the state transfer requests are executed in the DD architecture, the better memories are required for the SD architecture — an intuitive consequence of the fact that the most detrimental operations to gate fidelity in the DD architecture are moving requests.

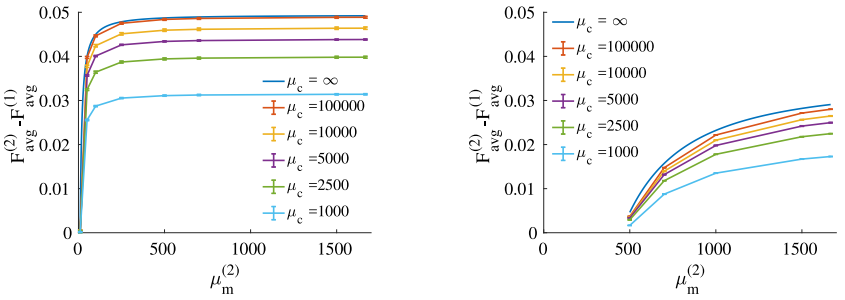
7. Simulation and numerical observations

Our goal in this section is to study the average gate and entanglement fidelities for the two architectures in a variety of settings in order to gain an understanding of regimes that are most suitable to each. We will also explore differences in manufacturing quality, and examine cases where it is preferable to use an SD design of better quality than a DD design with poorer quality. This question is especially relevant when cost-effectiveness is an important factor, as the DD design is expected to be the more expensive option (when comparing to an SD design of identical quality).

We use MATLAB to simulate the architectures and obtain the waiting time distributions for computational and entanglement jobs. We then use NetSquid [28] to simulate the storage of qubits according to the obtained waiting time distributions. NetSquid is a discrete-event network simulator for quantum information; it provides a hardware-validated model of the NV center in diamond platform, and we use this model to evaluate the gate and entanglement fidelities.

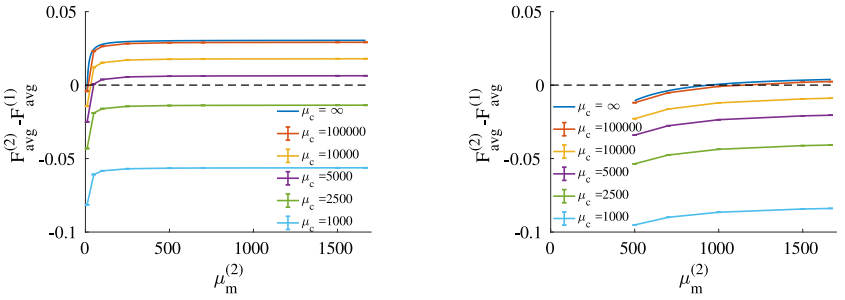
Our analytical results apply to the case when computational jobs have negligible processing time ($\mu_c = \infty$); thus, simulating the case where $\mu_c < \infty$ provides additional insight. When $\mu_c < \infty$, computational jobs may no longer be processed instantaneously. Thus, we require an additional rule in handling multiple computational requests in the queue. Motivated by the fact that entanglement generation with remote nodes is the most time-consuming operation for our architectures, we endow state transfer (moving) jobs non-preemptive priority over computational jobs, *i.e.*, when a moving job arrives while a computation is in progress, the former begins processing immediately after the completion of the computational request, even if other computational jobs were already in the queue prior to its arrival. In all other cases, jobs processed according to a FIFO policy.

For the following discussion, all rates are in terms of (# arrivals)/sec and (# jobs processed)/sec, unless otherwise specified. In all simulations, each run lasts for 10^5 s, and each data point is an average of five runs — a number that ensures sufficiently small error bars. We next motivate some of the parameter values used in the remainder of this section, many of which are inspired by the NV center in diamond platform. For state transfers within the SD architecture, we fix $\mu_m^{(1)}$ at 1667 Hz, since *cf.* Figure 4 in [19], a local swap to memory consumes $600\mu s$. State transfers within the DD architecture depend on the exact implementation of the transfer procedure across the inter-device interface. For the DD architecture, we often fix $\mu_m^{(2)} = 700$ Hz, as we expect the state transfer rate in this architecture to be between a third and a half of that of the SD architecture. Note that, according to [19,50], for the NV, this value is rather optimistic, since *e.g.*, a Bell-state measurement – an operation that is part of the state transfer gate sequence for the DD architecture – alone consumes 1 ms. Next, when exploring different values for the computation rate, we consider the variation in not only



(a) $\mu_e = 10$, $\lambda_e = 1$ (b) $\mu_e = 500$, $\lambda_e = 50$

Fig. 6. Average gate fidelity differences in two entanglement request arrival and generation rate regimes. For all simulations, $T_1^{(1)} = T_1^{(2)} = 0.00286$s and $T_2^{(1)} = T_2^{(2)} = 0.001$s; $\mu_m^{(1)} = 1667$ Hz, $\lambda_c = 150$ and $\lambda_m = 1000$.



(a) $\mu_e = 10$, $\lambda_e = 1$ (b) $\mu_e = 500$, $\lambda_e = 50$

Fig. 7. Average gate fidelity differences in two entanglement request arrival and generation rate regimes. For all simulations, $T_1^{(1)} = 10$s, $T_2^{(1)} = 0.01$s, $T_1^{(2)} = 2$s, and $T_2^{(2)} = 0.002$s; $\mu_m^{(1)} = 1667$ Hz, $\lambda_c = 150$ and $\lambda_m = 1000$.

gate duration, but also the possibility of more time-consuming atomic gate sequences that must not be interrupted by any other operations; using [18] as a guide, we set μ_c to values in the range $[10^3, 10^5]$. For T_1 and T_2 values we use [18] as a guide. Finally, when choosing parameter values for the request arrival rates, we ensure that both the SD and DD systems are stable in the number of outstanding entanglement requests.

Effects of device memory lifetimes on gate fidelity

Recall from our analytical evaluation that the DD architecture achieves higher gate fidelity than the SD architecture when both devices have the same T_1 and T_2 parameters characterizing their memory lifetime. This phenomenon can be observed in Fig. 6 for two different entanglement generation regimes: $\mu_e = 10$ corresponds to a quantum network setting, where the remote node(s) is(are) distant, while $\mu_e = 500$ represents closely-located quantum nodes, *e.g.*, as may be found in a quantum computing cluster. Observe that for the latter, the fidelity differences are less pronounced. This can be explained by the fact that faster entanglement generation rates are less detrimental to the SD architecture's gate fidelity than slower ones, as computational jobs wait less before being serviced.

Fabrication of networked quantum hardware is a complex task, and achieving comparable memory lifetimes between the SD and DD architectures may prove to be difficult. Interfacing the two processors that make up the DD architecture may introduce additional sources of noise or complicate the process of properly shielding qubits in order to maintain adequate memory lifetimes. As a result, it is possible that a high-quality SD design is both the more economical and functional choice, compared to a poorer-quality DD design. A potential instance of this is presented in Fig. 7, where the memory lifetimes of SD are five times those of the DD design. The advantages of the SD design are especially notable in the higher entanglement generation rate regime.

In Fig. 8, we explore the effect of manufacturing differences on the average gate fidelity further, focusing only on the analytical case ($\mu_c = \infty$). Here, we assume that $T_1^{(1)} > T_1^{(2)}$ and $T_2^{(1)} > T_2^{(2)}$. Each data point in the figure represents the difference between the SD and DD average gate fidelities for given $T_{1/2}^{(1/2)}$ values, with positive values representing cases where the SD performs better than the DD architecture. Note that not all combinations of T_1 and T_2 values are possible, as per the usual requirement that $T_1 > T_2$ for each architecture. From the figure, we observe that the T_2 time plays a

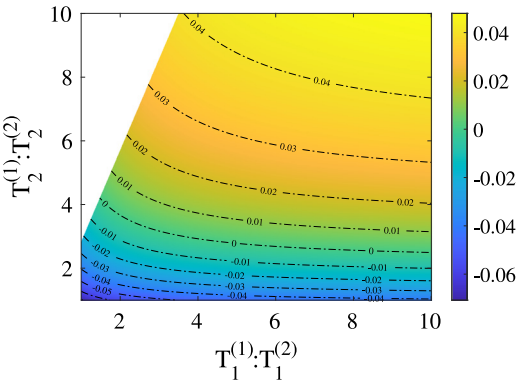


Fig. 8. Differences in average gate fidelity for the SD and DD architectures. For the latter, $T_1^{(2)}$ and $T_2^{(2)}$ are fixed at 0.00286s and 0.001s, respectively, while the ratios of the T_1 and T_2 times are varied. Here, $\mu_c = \infty$, $\lambda_e = 225$, $\mu_e = 500$, $\mu_m^{(1)} = 1667$ Hz, $\mu_m^{(2)} = 700$ Hz, and $\lambda_m = 1000$ to ensure stability.

significant role in improving gate fidelity: note in particular that for lower values of $T_2^{(1)}$, the SD architecture does not outperform the DD architecture, even for very high values of $T_1^{(1)}$.

Effects of processing rates on gate fidelity

Processing rates impact the amount of time needed to complete requests and consequently result in longer waiting times. Our analytical evaluation assumed that $\mu_c = \infty$ so that computational requests do not interfere with entanglement and moving requests. We use the computational job processing rate, μ_c, to capture the behavior of different applications, where smaller processing rates may correspond to computation requests performing a sequence of atomic gates or computation requiring "implicit" state transfers (see Appendix D.1). In both Figs. 6 and 7, we observe that more time-consuming computations are more hospitable to the SD architecture's gate fidelity. This phenomenon arises from the non-preemptive priority of moving jobs, which interrupt computation for longer periods in the DD architecture. To see how the individual average gate fidelities evolve as a function of μ_c, see Appendix F.

Effects of arrival rates on gate fidelity

The computation, entanglement, and moving request arrival rates impact the number of jobs that are issued to the system and consequently the probability that jobs block one another when non-zero processing time is assumed. Since no jobs can be processed in parallel on the SD architecture, this leads to an overall increase in waiting time for all requests. In contrast, computation jobs in the DD architecture are only blocked by moving jobs, while entanglement jobs may be processed in parallel to computation jobs. When considering the results in Figs. 6 and 7, one key observation is that both the entanglement request and moving request arrival rates impact the gate fidelity of computation jobs on the DD architecture. By increasing the arrival rate of entanglement requests, additional moving requests are submitted which leads to a decrease in the gate fidelity of computation requests (as further evidenced by Figs. F.14 and F.15 in Appendix F). In contrast, we observe an increase in the gate fidelity of the SD architecture here as the entanglement generation rate also increases, leading to a decrease in the amount of time that queued computation requests are blocked from processing.

Effects of moving entanglement

We now examine the effects of waiting time on the pre- and post-move entanglement fidelity in both architectures when they are realized with the NV center in diamond platform. Recall from Section 2.2 that the SD and DD architectures process moving requests in different ways. The SD architecture performs a sequence of local gates whereas the DD architecture must execute a network operation to transfer entanglement from the networking to the computing device (see Appendix D.2 for further details). Since quantum gates in both architectures are imperfect, it is important to highlight the differences in entanglement fidelity once a moving request has been processed. The *pre-move* entanglement fidelity is computed immediately before the transfer request is processed. *I.e.*, this measure incorporates the effects of waiting time both from needing to wait for the arrival of the transfer request, as well as the possible extra waiting time due an in-progress computation (recall that transfer requests have non-preemptive priority over computational jobs). The *post-move* entanglement fidelity is computed immediately after the entanglement has been moved from the networking to the computing component.

For the SD design, we are able to obtain the post-move fidelity in closed form (see Appendix E.2). We present it in Fig. 9, as a function of the moving request rate λ_m and the memory lifetimes of the architecture. As a reference, basic QKD demands entanglement fidelity of at least 0.81 [51]. Fig. 9 shows that, as expected, the moving request arrival rate (*i.e.*, the application's responsiveness to processing newly-generated entanglement) may be reduced if higher memory

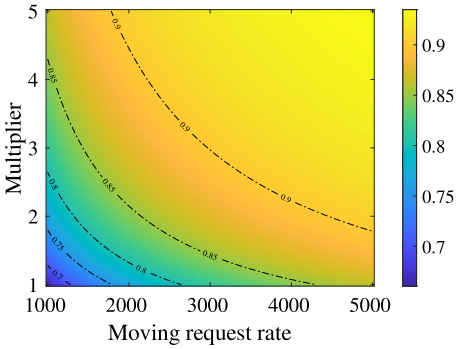
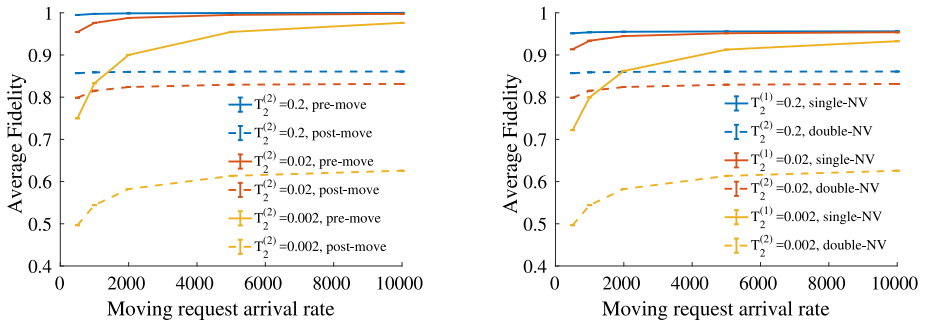


Fig. 9. Post-move fidelity for the SD architecture, as a function of the moving request rate λ_m. The characteristic memory times are varied at a constant ratio $T_1^{(1)} : T_2^{(1)}$, with initial values (*i.e.*, when the multiplier is set to one) of 0.00286s and 0.001s for $T_1^{(1)}$ and $T_2^{(1)}$, respectively.



(a) DD architecture pre- and post-move fidelities. (b) SD and DD architecture post-move fidelities.

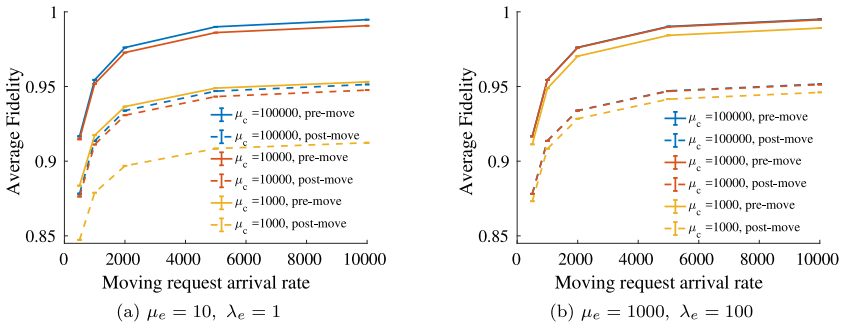
Fig. 10. Pre- and post-move fidelities for the DD architecture, (*a*), and post-move fidelities for the two architectures, (*b*). Here, $T_1^{(1/2)} = 2$s, $\lambda_e = 1$, $\mu_e = 10$, $\lambda_c = 150$, $\mu_c = 10^5$, $\mu_m^{(1)} = 1667$ Hz, and $\mu_m^{(2)} = 700$ Hz.

lifetimes are available. For the DD architecture, we obtain the entanglement fidelity via NetSquid. Fig. 10(a) presents its pre- and post-move fidelities for varying $T_2^{(2)}$ times, a parameter to which the post-move fidelity is highly sensitive. Indeed, we observe that the moving request arrival rate is not nearly as impactful to the fidelity as $T_2^{(2)}$. Fig. 10(b) presents a comparison of the two architectures' post-move fidelities; the SD design clearly outperforms the DD design for each of the T_2 values.

Fig. 11 shows the reduction between the pre-move and post-move entanglement fidelities for the SD architecture, in two entanglement generation regimes. It is immediately evident that in the high entanglement generation rate regime (*e.g.*, one that may be representative of a distributed quantum cluster setting), the pre- and post-move fidelities fair far better than in the low entanglement generation rate regime (*e.g.*, one that is more representative of a quantum network with distant nodes). A reason for this is that faster processing of entanglement requests in the SD design frees up processing time for computation. Consequently, there are fewer computation requests left in the queue, so that new entanglement requests can be processed more quickly as well.

Summary of findings

From our analysis and numerical observations, we find stark contrasts between the two architectures. On the one hand, when implemented with memories of identical quality, the DD design dominates in terms of gate fidelity. However, in a more practical scenario, wherein the DD design's more complex manufacturing would impair its memory lifetimes, the SD design can yield higher gate fidelities, and is more robust to longer computation times. Further, for present-day parameters, the SD design is more hospitable to the entanglement fidelity. The advantages of the SD design are especially evident in the high entanglement generation rate regime. We thus conclude that the DD design is more suitable for settings such as long-distance quantum communication, with lower entanglement generation rates and lighter computational



(a) $\mu_e = 10,\ \lambda_e = 1$ (b) $\mu_e = 1000,\ \lambda_e = 100$

Fig. 11. Pre-move and post-move fidelities for varying computational job processing rates in the SD architecture, as functions of the moving request arrival rate λ_m. Here, $T_1^{(1)} = 10$s and $T_2^{(1)} = 0.01$s; $\lambda_c = 150$ and $\mu_m^{(1)} = 1667$ Hz for all SD simulations and we assume the qubits are initially in a perfect entangled state.

demands. In contrast, the SD design is better suited for settings such as a distributed quantum computing cluster where high entanglement generation rates can be achieved and longer computations must be performed.

8. Conclusion

Quantum distributed applications impose quality constraints on the quantum states that they consume. When such applications are executed on architectures with physical limitations, such as imperfect gates or a limited amount of parallelism, resource contention can significantly impact performance, as some quantum states may be forced to wait in storage while others are being processed. In this work, we studied the effects of waiting times on the gate and entanglement fidelities for two distributed quantum architectures. We accomplished this by deriving formulas for average fidelity as a function of the waiting time distribution for a quantum state awaiting processing, as well as a noise model that governs quantum state evolution during storage. We obtained the waiting time distributions from the analysis of a Markov chain that models both of the quantum architectures in a regime where computation consumes a negligible amount of time; we later relaxed this assumption to study the effects of more time-consuming computation via simulation. We discovered that certain architecture implementations are more suitable for environments that are computation-heavy, while others are suitable for entanglement-heavy applications. Our average fidelity formulas are applicable in scenarios beyond those studied in this work, and may serve as a useful tool for performance evaluation experts.

Several extensions of our problem formulation are possible. First, we examined only two possible architecture implementations in this manuscript, but other realizations of distributed quantum architectures can be proposed. For instance, in the DD architecture, one could equip both of the devices with an interface to the outside world, so that both devices can perform local computation as well as remote entanglement generation. It is not entirely obvious what advantages one would gain in such a setup, and much like we have observed in the current work, the performance of such a system will depend on entanglement generation rates and the quality of the interface between the two devices (for transporting qubits from one to the other), as well as the application type (*e.g.*, computation-heavy vs. entanglement-heavy). Also, in the current work we assumed that each architecture has a single link to the outside world; an extension of the problem would be to consider multiple links, each associated with a different entanglement generation rate. Second, even with the two architectures examined in the current work, we have not studied all possible use cases. For instance, in the SD design, one could take further advantage of processor idle time by allowing computation to occur during entanglement generation, while the device is awaiting a heralding signal from the link. This would require the state of the (not yet entangled) qubit to be moved to a storage qubit, and then back again to the communication qubit; thus, for a rigorous analysis, one would have to account for how these state transfer operations would reflect on the final entanglement fidelity. Finally, from a modeling perspective, one could relax several of our assumptions, *e.g.*, that computational jobs consume zero time — something that we have so far only explored via simulation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported in part by the NWO ZK QSC Ada Lovelace Fellowship, The Netherlands. The authors thank Filip Rozpędek, Guus Avis, Francisco Ferreira da Silva, and David Maier for useful discussions and careful reading of an earlier version of the manuscript.

Appendix A. CTMC Analysis

A.1. Rate matrix

Define the following useful variables:

$$\alpha := \lambda_e + \mu_e, \quad \beta := \lambda_e + \lambda_m, \quad \text{and} \quad \gamma := \lambda_e + \mu_m. \tag{A.1}$$

The infinitesimal generator of the CTMC in Fig. 4 is given by

$$Q = \begin{bmatrix} B_{00} & B_{01} & \mathbf{0} & \cdots & & & \\ B_{10} & A_1 & A_2 & \mathbf{0} & \cdots & & \\ \mathbf{0} & A_0 & A_1 & A_2 & \mathbf{0} & \cdots & \\ \mathbf{0} & \mathbf{0} & A_0 & A_1 & A_2 & \mathbf{0} & \cdots \\ \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \end{bmatrix},$$

where

$$B_{00} = -\lambda_e, \quad B_{01} = \begin{bmatrix} \lambda_e & 0 & 0 \end{bmatrix}, \quad B_{10} = \begin{bmatrix} 0 \\ 0 \\ \mu_m \end{bmatrix},$$

$$A_0 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ \mu_m & 0 & 0 \end{bmatrix}, \quad A_1 = \begin{bmatrix} -\alpha & \mu_e & 0 \\ 0 & -\beta & \lambda_m \\ 0 & 0 & -\gamma \end{bmatrix}, \quad \text{and} \quad A_2 = \begin{bmatrix} \lambda_e & 0 & 0 \\ 0 & \lambda_e & 0 \\ 0 & 0 & \lambda_e \end{bmatrix},$$

and the $\mathbf{0}$'s are vectors or matrices of appropriate dimensions. The ergodicity condition for Markov chains whose generators have tridiagonal block structure are well-studied in literature, see, *e.g.*, [52]; we derive it here for completeness. The QBD process driven by Q is ergodic if and only if

$$\boldsymbol{\omega} A_2 \boldsymbol{e} < \boldsymbol{\omega} A_0 \boldsymbol{e}, \tag{A.2}$$

where $\boldsymbol{\omega}$ is the equilibrium distribution of the generator $A_0 + A_1 + A_2$ and $\boldsymbol{e}$ is a vector of all ones. To find $\boldsymbol{\omega}$, we use the relation

$$\boldsymbol{\omega}(A_0 + A_1 + A_2) = \mathbf{0},$$

or

$$\begin{bmatrix} \omega_1 & \omega_2 & \omega_3 \end{bmatrix} \begin{bmatrix} -\mu_e & \mu_e & 0 \\ 0 & -\lambda_m & \lambda_m \\ \mu_m & 0 & -\mu_m \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix},$$

which, along with the normalizing condition on ω_i, yield

$$\omega_1 = \frac{\lambda_m \mu_m}{\lambda_m \mu_m + \lambda_m \mu_e + \mu_e \mu_m}, \quad \omega_2 = \frac{\mu_e \mu_m}{\lambda_m \mu_m + \lambda_m \mu_e + \mu_e \mu_m}, \quad \text{and} \quad \omega_3 = \frac{\lambda_m \mu_e}{\lambda_m \mu_m + \lambda_m \mu_e + \mu_e \mu_m}.$$

Thus, after defining $D := \lambda_m \mu_m + \lambda_m \mu_e + \mu_e \mu_m$ and with the assumption that $D > 0$ since all rates are positive, (A.2) becomes

$$\frac{1}{D}\begin{bmatrix} \lambda_m \mu_m & \mu_e \mu_m & \lambda_m \mu_e \end{bmatrix} \begin{bmatrix} \lambda_e & 0 & 0 \\ 0 & \lambda_e & 0 \\ 0 & 0 & \lambda_e \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} < \frac{1}{D}\begin{bmatrix} \lambda_m \mu_m & \mu_e \mu_m & \lambda_m \mu_e \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ \mu_m & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix},$$

$$\lambda_e \begin{bmatrix} \lambda_m \mu_m & \mu_e \mu_m & \lambda_m \mu_e \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} < \begin{bmatrix} \lambda_m \mu_m & \mu_e \mu_m & \lambda_m \mu_e \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \mu_m \end{bmatrix},$$

$$\lambda_e(\lambda_m \mu_m + \mu_e \mu_m + \lambda_m \mu_e) < \lambda_m \mu_e \mu_m,$$

or, written another way,

$$\frac{1}{\lambda_e} > \frac{1}{\mu_e} + \frac{1}{\lambda_m} + \frac{1}{\mu_m}. \tag{A.3}$$

Intuitively, (A.3) indicates that for ergodicity, the average time between entanglement request arrivals must exceed the average processing times summed over all three stages (entanglement generation, waiting for a moving request to arrive, and performing the moving operation).

Henceforth, assume that (A.3) is satisfied. Next, we obtain the rate matrix of the generator. Note that A_0 is of rank 1; we may rewrite it as follows:

$$A_0 = \begin{bmatrix} 0 \\ 0 \\ \mu_m \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}.$$

By the results in [47], this means that the rate matrix may be computed explicitly; it is given by

$$R = -A_2(A_1 + A_2 \boldsymbol{e} \begin{bmatrix} 1 & 0 & 0 \end{bmatrix})^{-1}. \tag{A.4}$$

After some algebra, we obtain

$$R = \frac{\lambda_e}{\lambda_m \mu_e \mu_m} \begin{bmatrix} \beta\gamma & \gamma\mu_e & \mu_e\lambda_m \\ \lambda_e(\gamma + \lambda_m) & \gamma\mu_e & \mu_e\lambda_m \\ \lambda_e\beta & \lambda_e\mu_e & \mu_e\lambda_m \end{bmatrix}, \tag{A.5}$$

which matches the explicit rate matrix computation in [53] for the $M/HYPO_K/1$ queue when $K = 3$.

A.2. Probability that a computational job must wait

Our goal here is to compute $\sum_{N=1}^{\infty} \pi_{N/1}$ and $\sum_{N=1}^{\infty} \pi_{N/3}$. To derive these quantities, we use the global balance principle to "cut" the chain three different ways (the first cut isolates the states $N/3$, the second isolates the states 0 and $N/1$, and the third isolates the states $N/2$), obtaining the following balance equations (below, $\pi_{N/2}$ is the stationary probability of state $N/2$):

$$\mu_m \sum_{N=1}^{\infty} \pi_{N/3} = \lambda_m \sum_{N=1}^{\infty} \pi_{N/2}, \tag{A.6}$$

$$\lambda_m \sum_{N=1}^{\infty} \pi_{N/2} = \mu_e \sum_{N=1}^{\infty} \pi_{N/1}, \tag{A.7}$$

$$\mu_e \sum_{N=1}^{\infty} \pi_{N/1} = \mu_m \sum_{N=1}^{\infty} \pi_{N/3}. \tag{A.8}$$

We can rewrite (A.6) as follows:

$$\mu_m \sum_{N=1}^{\infty} \pi_{N/3} = \lambda_m \left(1 - \sum_{N=1}^{\infty} \pi_{N/1} - \sum_{N=1}^{\infty} \pi_{N/3} - \pi_0\right), \tag{A.9}$$

where π_0 is the stationary probability of state 0, and using (A.8), (A.9) becomes

$$\mu_m \sum_{N=1}^{\infty} \pi_{N/3} = \lambda_m \left(1 - \frac{\mu_m}{\mu_e} \sum_{N=1}^{\infty} \pi_{N/3} - \sum_{N=1}^{\infty} \pi_{N/3} - \pi_0\right), \tag{A.10}$$

$$\left(\mu_m + \frac{\lambda_m \mu_m}{\mu_e} + \lambda_m\right) \sum_{N=1}^{\infty} \pi_{N/3} = \lambda_m (1 - \pi_0), \tag{A.11}$$

$$\sum_{N=1}^{\infty} \pi_{N/3} = \frac{\lambda_m (1 - \pi_0)}{\mu_m + \frac{\lambda_m \mu_m}{\mu_e} + \lambda_m} = \frac{\lambda_m (1 - \pi_0) \mu_e}{\mu_m \mu_e + \lambda_m \mu_m + \lambda_m \mu_e}. \tag{A.12}$$

To obtain π_0, we use the balance equations of the QBD process along with the normalizing condition:

$$\begin{bmatrix} \pi_0 & \boldsymbol{\pi_1} \end{bmatrix} \begin{bmatrix} B_{00} & B_{01} \\ B_{10} & A_1 + RA_0 \end{bmatrix} = \begin{bmatrix} 0 & \mathbf{0} \end{bmatrix}, \tag{A.13}$$

$$\pi_0 + \boldsymbol{\pi_1}(I - R)^{-1} \boldsymbol{e} = 1, \tag{A.14}$$

where $\boldsymbol{\pi_1} \equiv \begin{bmatrix} \pi_{1/1} & \pi_{1/2} & \pi_{1/3} \end{bmatrix}$; B_{00}, B_{01}, B_{10}, A_1, and A_0 are blocks of the CTMC generator matrix defined in Appendix A.1, R is the rate matrix of the QBD process derived in Appendix A.1, $\boldsymbol{e}$ is a vector of all ones, and I is an identity matrix of the same dimensions as R.

From (A.13) and (A.14), we obtain

$$\pi_0 = \frac{\Delta}{\lambda_m \mu_e \mu_m}, \quad \pi_{1/1} = \frac{\lambda_e(\lambda_e + \lambda_m)(\lambda_e + \mu_m)\Delta}{(\lambda_m \mu_e \mu_m)^2}, \quad \pi_{1/2} = \frac{\lambda_e(\lambda_e + \mu_m)\Delta}{\lambda_m^2 \mu_e \mu_m^2}, \quad \pi_{1/3} = \frac{\lambda_e \Delta}{\lambda_m \mu_e \mu_m^2}, \tag{A.15}$$

where $\Delta \equiv (\lambda_m\mu_e\mu_m - \lambda_e\lambda_m\mu_e - \lambda_e\lambda_m\mu_m - \lambda_e\mu_e\mu_m)$. Note that $\Delta > 0$ follows directly from the ergodicity of the chain. Using (A.12), (A.15) and the definition of Δ, we obtain

$$\sum_{N=1}^{\infty}\pi_{N/3} = \frac{\lambda_m\left(1-\frac{\lambda_m\mu_e\mu_m-\lambda_e\lambda_m\mu_e-\lambda_e\lambda_m\mu_m-\lambda_e\mu_e\mu_m}{\lambda_m\mu_e\mu_m}\right)\mu_e}{\mu_m\mu_e+\lambda_m\mu_m+\lambda_m\mu_e} = \frac{\lambda_e}{\mu_m}. \tag{A.16}$$

Using (A.8), we have

$$\sum_{N=1}^{\infty}\pi_{N/1} = \frac{\mu_m}{\mu_e}\sum_{N=1}^{\infty}\pi_{N/3} = \frac{\mu_m}{\mu_e}\frac{\lambda_e}{\mu_m} = \frac{\lambda_e}{\mu_e}. \tag{A.17}$$

Appendix B. Proof of Proposition 1

Proof. Recall that the SD architecture's average gate fidelity is given by

$$\begin{aligned} F^{(1)}_{avg}(\mathcal{N}_t, G) &= \int_0^{\infty} F(\mathcal{N}_t, G) f_{W_1}(t)dt \\ &= \int_0^{\infty} F(\mathcal{N}_t, G)\left(\lambda_e e^{-\mu_e t} + \lambda_e e^{-\mu_m^{(1)}t} + \left(1-\frac{\lambda_e}{\mu_e}-\frac{\lambda_e}{\mu_m^{(1)}}\right)\delta(t)\right)dt \\ &= \lambda_e\int_0^{\infty} F(\mathcal{N}_t, G)\left(e^{-\mu_e t} + e^{-\mu_m^{(1)}t}\right)dt + 1 - \frac{\lambda_e}{\mu_e} - \frac{\lambda_e}{\mu_m^{(1)}}, \end{aligned} \tag{B.1}$$

while for the DD architecture,

$$\begin{aligned} F^{(2)}_{avg}(\mathcal{N}_t, G) &= \int_0^{\infty} F(\mathcal{N}_t, G) f_{W_2}(t)dt \\ &= \int_0^{\infty} F(\mathcal{N}_t, G)\left(\lambda_e e^{-\mu_m^{(2)}t} + \left(1-\frac{\lambda_e}{\mu_m^{(2)}}\right)\delta(t)\right)dt \\ &= \lambda_e\int_0^{\infty} F(\mathcal{N}_t, G) e^{-\mu_m^{(2)}t}dt + 1 - \frac{\lambda_e}{\mu_m^{(2)}}. \end{aligned} \tag{B.2}$$

Next, note that for a function $0 \le g(t) \le 1$ and $x, y > 0$ with $x \ge y$,

$$\int_0^{\infty} g(t)e^{-xt}dt - \frac{1}{x} \ge \int_0^{\infty} g(t)e^{-yt}dt - \frac{1}{y}, \tag{B.3}$$

since

$$\int_0^{\infty} g(t)(e^{-yt}-e^{-xt})dt \le \int_0^{\infty}(e^{-yt}-e^{-xt})dt = \frac{1}{y}-\frac{1}{x}.$$

Since $\mu_m^{(2)} \ge \mu_e$, it follows from (B.3) that

$$F^{(2)}_{avg} \ge \lambda_e\int_0^{\infty} F(\mathcal{N}_t, G)e^{-\mu_e t}dt + 1 - \frac{\lambda_e}{\mu_e}. \tag{B.4}$$

Call the quantity on the right-hand of (B.4) $\hat{F}$. Note that

$$\begin{aligned} \hat{F} - F^{(1)}_{avg} &= \lambda_e\int_0^{\infty} F(\mathcal{N}_t, G)e^{-\mu_e t}dt + 1 - \frac{\lambda_e}{\mu_e} - \left(\lambda_e\int_0^{\infty} F(\mathcal{N}_t, G)\left(e^{-\mu_e t} + e^{-\mu_m^{(1)}t}\right)dt + 1 - \frac{\lambda_e}{\mu_e} - \frac{\lambda_e}{\mu_m^{(1)}}\right) \\ &= \frac{\lambda_e}{\mu_m^{(1)}} - \lambda_e\int_0^{\infty} F(\mathcal{N}_t, G)e^{-\mu_m^{(1)}t}dt \ge 0, \end{aligned} \tag{B.5}$$

where the last inequality follows from the fact that $F(\mathcal{N}_t, G) \le 1$, for any t. By (B.4) and (B.5), we conclude that $F^{(2)}_{avg} \ge F^{(1)}_{avg}$. □

Appendix C. Proof of bound (29)

To prove (29), we first prove the following useful facts.

Lemma 2. *For any reals a, x, and y all greater than zero, if $x > y$, then*

$$\frac{x}{ax+1} > \frac{y}{ay+1}. \tag{C.1}$$

Proof. (Lemma 2) Assume for a contradiction that (C.1) is not true. Then, it must be that

$$\frac{x}{ax+1} \leq \frac{y}{ay+1}, \implies x(ay+1) \leq y(ax+1), \implies x \leq y,$$

which contradicts the hypothesis. □

Lemma 3. *For any reals a, b, and x all greater than zero,*

$$\frac{1}{a(ax+1)} + \frac{1}{b(bx+1)} < \frac{2}{\frac{\sqrt{2}ab}{a+b}\left(\frac{\sqrt{2}ab}{a+b}x+1\right)}. \tag{C.2}$$

Proof. (Lemma 3) Let $c = \sqrt{2}ab/(a+b)$. Then (C.2) is equivalent to

$$\frac{(a^2+b^2)x + a + b}{ab(ax+1)(bx+1)} < \frac{2}{c\,(cx+1)}, \tag{C.3}$$

$$((a^2+b^2)x + a + b)c\,(cx+1) < 2ab(ax+1)(bx+1), \tag{C.4}$$

$$(a^2+b^2)c^2x^2 + \left((a+b)c + a^2 + b^2\right)cx + (a+b)c < 2ab(abx^2 + (a+b)x + 1). \tag{C.5}$$

Let us verify (C.5) by comparing the coefficients of terms x^j, $j = 2, 1, 0$, on each side. First, consider the coefficients of x^2: on the left-hand side of (C.5), we have

$$(a^2+b^2)c^2 = (a^2+b^2)\frac{2a^2b^2}{(a+b)^2}, \tag{C.6}$$

and it is easy to see that this quantity is less than $2a^2b^2$, the coefficient of x^2 on the right side of (C.5). Similarly, for the coefficient of x on the left hand side of (C.5), we have

$$\left((a+b)c + a^2 + b^2\right)c = \left(\sqrt{2}ab + a^2 + b^2\right)\frac{\sqrt{2}ab}{a+b}, \tag{C.7}$$

which is less than $2ab(a+b)$, the coefficient of x on the right side of (C.5). Finally, the constant term on the left side of (C.5) is $(a+b)c = \sqrt{2}ab$, while the constant term on the right is larger $(2ab)$. □

Proof. (Proposition 2) For this proof, it is useful to keep in mind that T_1 is always greater than T_2 for any architecture. From this fact, and by Lemma 2, it follows that to satisfy (28), it suffices to ensure that

$$\frac{1}{\mu_e} + \frac{1}{\mu_m^{(1)}} - \frac{T_2^{(1)}}{\mu_e T_2^{(1)} + 1} - \frac{T_2^{(1)}}{\mu_m^{(1)} T_2^{(1)} + 1} < \frac{1}{\mu_m^{(2)}} - \frac{T_1^{(2)}}{\mu_m^{(2)} T_1^{(2)} + 1}, \tag{C.8}$$

which we obtained by substituting each $T_1^{(1)}$ on the left-hand side of (28) with $T_2^{(1)}$ and each $T_2^{(2)}$ with $T_1^{(2)}$ on the right-hand side of (28). Next, (C.8) is equivalent to

$$\frac{1}{\mu_e(\mu_e T_2^{(1)} + 1)} + \frac{1}{\mu_m^{(1)}(\mu_m^{(1)} T_2^{(1)} + 1)} < \frac{1}{\mu_m^{(2)}(\mu_m^{(2)} T_1^{(2)} + 1)}. \tag{C.9}$$

To ensure (C.9) holds, it is sufficient to have

$$\frac{2}{\frac{\sqrt{2}\mu_e\mu_m^{(1)}}{\mu_e+\mu_m^{(1)}}\left(\frac{\sqrt{2}\mu_e\mu_m^{(1)}}{\mu_e+\mu_m^{(1)}} T_2^{(1)} + 1\right)} < \frac{1}{\mu_m^{(2)}(\mu_m^{(2)} T_1^{(2)} + 1)}. \tag{C.10}$$

Solving (C.10) for $T_2^{(1)}$ yields (29). □

Appendix D. Nitrogen-vacancy center in diamond

D.1. Overview

Here, we provide a brief overview of the Nitrogen-Vacancy (NV) center in diamond platform and its characteristics as relevant to our problem. For additional information, we refer the reader to, for example, [18]. The NV center in diamond platform is a few-qubit (at present, no more than 10 [54]) quantum processor capable of executing arbitrary quantum gates and measurements and has demonstrated entanglement establishment over a distance of 1.3 km [24]. This hardware has also been used to demonstrate key quantum network protocols required for long-distance networking such as entanglement swapping and distillation [19,55]. Qubits in the NV center in diamond platform may be divided

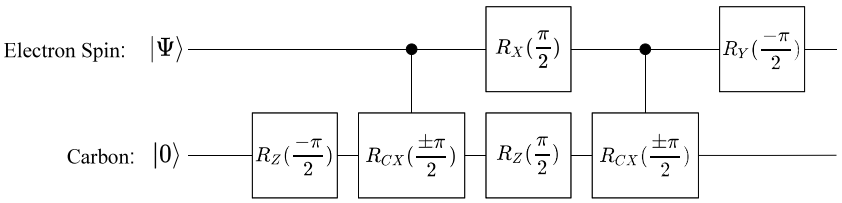


Fig. D.12. Quantum circuit for moving a quantum state from the NV into carbon storage. Each gate applies noise onto the qubits it interacts with.

into two types, *communication qubits* and *storage qubits*. Communication qubits are those that are equipped with optical interfaces, allowing them to establish entanglement with communication qubits in other quantum processors while storage qubits, on the other hand, are only capable of storing quantum states and having quantum gates applied to them. In the particular case of the NV center in diamond, the set of quantum gates that can be applied to storage qubits is limited to a time-dependent rotation about the Z axis, while arbitrary quantum gates may be applied to the communication qubit.

Interactions between qubits in the NV center in diamond platform are mediated by an electronic spin (the NV) that acts as a communication qubit. The remaining qubits are C^{13} spins that act as storage qubits which are magnetically coupled to the electronic spin. This property of NV in diamond hardware restricts the parallelism of quantum gates on different qubits as multiple quantum gates may not be applied to the NV at the same time, resulting in serial execution of quantum gates. Furthermore, most quantum gates may only be performed on quantum states held by the NV qubit, while C^{13} spins may only be initialized and undergo Z-rotations. This means that the application of a quantum gate to a state held by a storage qubit requires exchanging the quantum state of the NV with said storage qubit.

One implication of these physical restrictions on computation with the NV center in diamond is that for our model of the single-NV architecture (Section 3), computational jobs may require state transfers before they can be serviced. Specifically, such a situation may arise whenever the system is awaiting a state transfer request (to move the state of a newly-entangled qubit from the NV to a storage qubit), while one or more computation requests are in the queue. Recall that during such "idle" periods, according to our modeling framework computation is allowed in the SD architecture. However, depending on the type of computation that is being requested (see discussion above), the processor may need to perform a state transfer prior to the computation (NV $\rightarrow$ storage qubit) to free up the NV qubit, perform the computation (recall that when $\mu_c = \infty$, all computational jobs may be processed instantaneously), and finally, perform another state transfer (storage qubit $\rightarrow$ NV) to move the state of the entangled qubit back to the NV. One may wonder: why perform the latter move when the state of the entangled qubit must eventually be moved to storage? The reason is that future computation requests may require the use of the entangled qubit in the NV. On the other hand, it may be possible that all computation requests involve the entangled qubit in the NV, in which case the extra transfers are not required. Since we have no knowledge of the computational request requirements a priori, we simply assume within our model that "implicit" state transfers are performed when necessary, and that they also (akin to computation requests) consume a negligible amount of time. In a manner, our simulations in Section 7 relax this assumption by varying computation request processing rates. Applications that infrequently shuffle quantum states between qubits correspond to the case where μ_c is very large while applications that move quantum states more frequently correspond to the case when μ_c is smaller and comparable to μ_m.

In addition to the limitations on quantum gates, the NV qubit is the sole communication qubit that may be equipped with an optical interface for establishing entanglement with qubits in other quantum processors. This further restricts parallelism of operations as quantum gates may not be applied to any qubits when the processor is being used to establish entanglement. Studies on experimental realizations of such hardware have shown that the fidelity and the rate at which entangled states may be established between such processors decrease as the distance over which entanglement must be established increases [18,56]. Combined with existing challenges in emitted photon collection [50], this means that establishing entanglement will dominate the execution time of applications and incur additional latency for performing local quantum gates between qubits. Furthermore, the process of establishing entanglement introduces noise on quantum states that are stored in the remaining qubits of the system [57], which can severely reduce the fidelity of stored states.

D.2. Transferring quantum states

D.2.1. Single-device NV

Our NetSquid implementation for simulating the transfer of a quantum state from the NV electron spin qubit into carbon storage qubit is performed through a sequence of gates shown in the circuit of Fig. D.12 [55].

Here, the electron spin is in some quantum state $|\Psi\rangle$ and the carbon storage is initialized to the state $|0\rangle$. Gates are applied to each qubit in order from left to right and lines from the NV to a gate on the carbon denote a controlled quantum gate. For reference, the quantum gates used in our simulations are defined as

$$R_X(\theta) = \begin{bmatrix} \cos(\frac{\theta}{2}) & -i\sin(\frac{\theta}{2}) \\ -i\sin(\frac{\theta}{2}) & \cos(\frac{\theta}{2}) \end{bmatrix}, \quad R_Y(\theta) = \begin{bmatrix} \cos(\frac{\theta}{2}) & -\sin(\frac{\theta}{2}) \\ \sin(\frac{\theta}{2}) & \cos(\frac{\theta}{2}) \end{bmatrix}, \quad R_Z(\theta) = \begin{bmatrix} e^{-i\frac{\theta}{2}}0 & \\ 0 & e^{i\frac{\theta}{2}} \end{bmatrix},$$

Table D.2
Depolarizing parameters for gates in the NV hardware. We remark that no two NV devices are exactly identical. Individual values have not been realized simultaneously for producing entanglement which would allow a direct comparison to simulation. We thus focus on simulation parameters that enable a comparison to entanglement generation hardware and provide references to motivations for our chosen values.

	Qubits depolarized	p_G NV	p_G Carbon
Electron initialization [29]	Electron	0.02	–
Electron $R_X(\theta)$ [55]	–	–	–
Electron $R_Y(\theta)$ [55]	–	–	–
Carbon initialization [54]	Carbon	–	0.006/4
Carbon $R_Z(\theta)$ [58]	Carbon	–	0.001/3
Electron-Carbon $R_{CX}(\pm\theta)$ [55]	Electron and Carbon	0.005	0.005
Electron-Carbon $R_{CY}(\pm\theta)$ [55]	Electron and Carbon	0.005	0.005

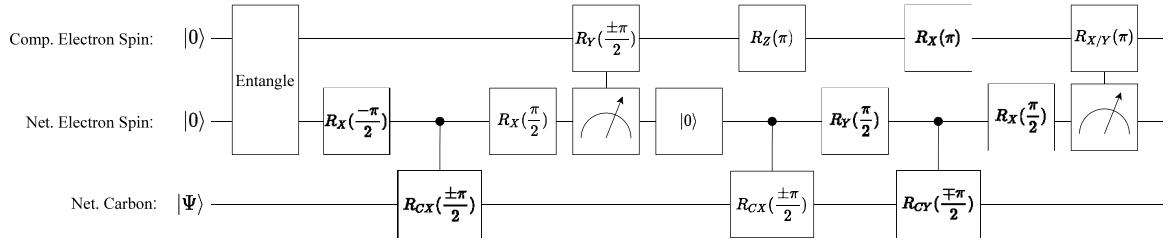


Fig. D.13. Quantum circuit for moving a quantum state from the networking device electron spin into the computing device electron spin.

$$R_{CX}(\pm\theta) = |0\rangle\langle 0| \otimes R_X(\theta) + |1\rangle\langle 1| \otimes R_X(-\theta) = \begin{bmatrix} \cos(\frac{\theta}{2}) & -i\sin(\frac{\theta}{2}) & 0 & 0 \\ -i\sin(\frac{\theta}{2}) & \cos(\frac{\theta}{2}) & 0 & 0 \\ 0 & 0 & \cos(\frac{\theta}{2}) & i\sin(\frac{\theta}{2}) \\ 0 & 0 & i\sin(\frac{\theta}{2}) & \cos(\frac{\theta}{2}) \end{bmatrix},$$

$$R_{CY}(\pm\theta) = |0\rangle\langle 0| \otimes R_Y(\theta) + |1\rangle\langle 1| \otimes R_Y(-\theta) = \begin{bmatrix} \cos(\frac{\theta}{2}) & -\sin(\frac{\theta}{2}) & 0 & 0 \\ \sin(\frac{\theta}{2}) & \cos(\frac{\theta}{2}) & 0 & 0 \\ 0 & 0 & \cos(\frac{\theta}{2}) & \sin(\frac{\theta}{2}) \\ 0 & 0 & -\sin(\frac{\theta}{2}) & \cos(\frac{\theta}{2}) \end{bmatrix},$$

where $\otimes$ denotes the tensor (Kronecker) product of two matrices.

In NetSquid, gates in the NV center in diamond platform are modeled as the application of the perfect gate G after applying time-independent depolarizing noise $\mathcal{D}_G$ that is parameterized by an associated depolarizing probability p_G depending on the gate G being applied. For the gate sequence in Fig. D.12, the depolarizing parameters for each gate are summarized in Table D.2.

As an example, consider applying the $R_Z(\frac{\pi}{2})$ to some state $\rho = |\psi\rangle\langle\psi|$ in a carbon qubit. The new state is

$$\mathcal{D}_G\left(R_Z\left(\frac{\pi}{2}\right)\rho R_Z^\dagger\left(\frac{\pi}{2}\right)\right)$$

using depolarizing probability $p_{R_Z} = \frac{0.001}{3}$ for $\mathcal{D}_G$.

D.2.2. Double-device NV

Our NetSquid implementation for simulating quantum state transfer from the NV electron spin of the networking device to the NV electron spin of the computing device expands upon the gate sequence in Fig. D.12 and makes use of quantum teleportation [15]. Once entanglement that was originally held by the networking device electron spin has been moved to the networking device carbon storage (using the gate sequence from Fig. D.12), the following gate sequence Fig. D.13 is performed in order to transfer the state to the computing device electron spin [19].

Here, the "Entangle" box represents establishing entanglement between the electron spins of the networking device and the computing device, the $|0\rangle$ box represents initializing the networking device electron spin to the $|0\rangle$ state, and boxes containing arcs with an arrow represent measuring the qubit in the standard (Z) basis. The "Entangle" operation instantly establishes a perfect entangled state between the electron spins of the computing device and the networking device, thus giving an optimistic evaluation of the state transfer process in the DD architecture. Connections between measurement boxes and a gate on the computing electron spin represent conditional execution of the gate based on the measurement result. In the first case, measuring $|0\rangle$ on the networking electron spin means we perform a $R_Y(\pi)$ gate on the computing NV while measuring $|1\rangle$ on the networking NV means we perform a $R_Y(-\pi)$ gate. Similarly for the second instance, here we perform a $R_X(\pi)$ gate on the computing electron spin if we measure $|0\rangle$ on the networking NV and a $R_Y(\pi)$ gate if we measure $|1\rangle$ on the networking electron spin.

Appendix E. Fidelity derivations

E.1. Alternative average gate fidelity derivation for noise channels

For completeness, we present an alternative, self-contained derivation of the average gate fidelity that depends only on a number of well-known tricks in quantum information. Our proof is based on the Choi–Jamiolkowski theorem establishing a duality between quantum states and quantum channels. First, we will make use of the notion of the Choi state of a quantum channel. Here, it will be sufficient to note that for channels $\mathcal{N} : \mathcal{S}_A \to \mathcal{S}_A$ where $\mathcal{S}_A$ denotes the state of a quantum system A, the Choi state for $\mathcal{N}$ is defined (see *e.g.*, [59]) as

$$\tau_{\mathcal{N}} = \mathcal{N} \otimes \mathbb{I}_{A'} \left(|\Phi_{AA'}\rangle\langle\Phi_{AA'}| \right) \ , \tag{E.1}$$

where

$$|\Phi_{AA'}\rangle = \frac{1}{\sqrt{d_A}} \sum_{j=1}^{d_A} |j\rangle_A |j\rangle_{A'} \tag{E.2}$$

is the maximally entangled state on A and an identical quantum system A'. We will also make use of the partial transpose operation Γ. For any operator $M_{AA'} \in \mathbb{C}^{d_A \times d_A} \otimes \mathbb{C}^{d_A \times d_A}$ the partial transpose is defined as

$$M_{AA'}^{\Gamma} = \left(\sum_{ijk\ell} c_{ij}^{k\ell} |i\rangle\langle j|_A \otimes |k\rangle\langle \ell|_{A'} \right)^{\Gamma} = \sum_{ijk\ell} c_{ij}^{k\ell} |i\rangle\langle j|_A \otimes (|k\rangle\langle \ell|_{A'})^T \ , \tag{E.3}$$

which corresponds to taking the transpose on A', but not on A.

We first establish the following lemma.

Lemma 4. *Let $\mathcal{N}_t : \mathcal{S}_A \to \mathcal{S}_A$ denote a noisy quantum channel depending on time t. For any quantum gate G on system A, we have*

$$F(\mathcal{N}_t, G) = \frac{d_A}{d_{\text{sym}}} \text{Tr}\left[\Pi_{\text{sym}} \tau_{\mathcal{N}_t}^{\Gamma} \right] \ , \tag{E.4}$$

where d_A is the dimension of the quantum system A, Π_{sym} is the projector onto the symmetric subspace of $\mathbb{C}^{d_A \times d_A} \otimes \mathbb{C}^{d_A \times d_A}$, d_{sym} is the dimension of said symmetric subspace, and $\tau_{\mathcal{N}_t}$ is the Choi state of $\mathcal{N}_t$.

Proof. Using the Choi–Jamiolkowski isomorphism (see *e.g.*, [59]), we can rewrite

$$\langle\Psi|G^\dagger \mathcal{N}_t(G|\Psi\rangle\langle\Psi|G^\dagger)G|\Psi\rangle = \text{Tr}\left[G|\Psi\rangle\langle\Psi|G^\dagger \mathcal{N}_t \left(G|\Psi\rangle\langle\Psi|G^\dagger \right) \right] \tag{E.5}$$

$$= d_A \text{Tr}\left[G|\Psi\rangle\langle\Psi|G^\dagger \otimes \left(G|\Psi\rangle\langle\Psi|G^\dagger \right)^T \tau_{\mathcal{N}_t} \right] \tag{E.6}$$

$$= d_A \text{Tr}\left[\left(G|\Psi\rangle\langle\Psi|G^\dagger \right)^{\otimes 2} \tau_{\mathcal{N}_t}^{\Gamma} \right] \ . \tag{E.7}$$

Using the definition for the average gate fidelity (12) and (E.7), we can then write

$$F(\mathcal{N}_t, G) = d_A \text{Tr}\left[\left(\int d\Psi \left(G|\Psi\rangle\langle\Psi|G^\dagger \right)^{\otimes 2} \right) \tau_{\mathcal{N}_t}^{\Gamma} \right] \tag{E.8}$$

$$= \frac{d_A}{d_{\text{sym}}} \text{Tr}\left[\Pi_{\text{sym}} \tau_{\mathcal{N}_t}^{\Gamma} \right] \ , \tag{E.9}$$

where we have used the fact that

$$\int d\Psi \left(G|\Psi\rangle\langle\Psi|G^\dagger \right)^{\otimes 2} = \int d\Psi \, |\Psi\rangle\langle\Psi|^{\otimes 2} \tag{E.10}$$

$$= \frac{\Pi_{\text{sym}}}{d_{\text{sym}}} \ . \quad \square \tag{E.11}$$

Given the little lemma above, we can now readily evaluate the gate fidelity for any channel of interest in two steps: first, we need an expression for the projector Π_{sym} onto the symmetric subspace. It is well known (see *e.g.*, [60]) that the full set of so-called mutually unbiased basis in dimension $d = 2^n$ forms a 2-design, *i.e.*,

$$\int d\Psi \, |\Psi\rangle\langle\Psi|^{\otimes 2} = \frac{1}{d(d+1)} \sum_{\theta} \sum_{j} |j_\theta\rangle\langle j_\theta| \ , \tag{E.12}$$

$$= \frac{2\Pi_{\text{sym}}}{d(d+1)}, \tag{E.13}$$

where the sum extends over bases indexed by θ and $|j_\theta\rangle$ denotes the jth basis state of basis θ. For $d = 2$, *i.e.*, a single qubit, these bases are simply the eigenbases of the operators X, Z and Y defined in Section 2.2. Using (E.11), this allows us to write

$$\Pi_{\text{sym}} = \frac{1}{2} \sum_{\theta \in \{X,Z,Y\}} \sum_{j} |j_\theta\rangle\langle j_\theta| \,, \tag{E.14}$$

where $d_{\text{sym}} = 3$. In such small dimensions, it is also easy to write

$$\Pi_{\text{sym}} = |\Phi_{00}\rangle\langle\Phi_{00}| + |\Phi_{01}\rangle\langle\Phi_{01}| + |\Phi_{10}\rangle\langle\Phi_{10}| \,, \tag{E.15}$$

where $|\Phi_{ab}\rangle = \mathbb{I} \otimes X^a Z^b |\Phi\rangle$ denotes the first three Bell states (excluding the singlet). Second, we need to compute the Choi states of the noise channels defined in Section 2.2.2, which can readily be achieved by using (E.1).

E.2. Average post-move entanglement fidelity in single-NV architecture

Using Lemma 4 we may also obtain analytic expressions for the entanglement fidelity of the state that was moved into memory.

We may analytically compute the gate fidelity for the move to memory gate sequence in Fig. D.12 as

$$\begin{aligned} F^{pm} = \frac{1}{6}(3 - 6p_{R_X} + (p_{init} - 1)(p_{R_Z} - 1)^2(p_{R_{CX}} - 1)^2(2p_{R_X} - 1)e^{-\frac{t}{T_1}} \\ + (2 + p_{init}(p_{R_Z} - 1) - p_{R_Z})(p_{R_Z} - 1)(p_{R_{CX}} - 1)^3(4p_{R_X} - 1)e^{-\frac{t}{T_2}}), \end{aligned} \tag{E.16}$$

where p_{init}, p_{R_Z}, p_{R_X} and $p_{R_{CX}}$ are the depolarizing probabilities for Carbon Initialization, Carbon R_Z rotations, Electron R_X rotations, and Electron-Carbon R_{CX} respectively. We may now obtain the average post-move gate fidelity by integrating over the waiting time distribution of move requests,

$$\begin{aligned} F^{pm}_{avg} &= \int_0^\infty \lambda_m e^{-\lambda_m t} \tfrac{1}{6}(3 - 6p_{R_X} + (p_{init} - 1)(p_{R_Z} - 1)^2(p_{R_{CX}} - 1)^2(2p_{R_X} - 1)e^{-\frac{t}{T_1}} \\ &\quad + (2 + p_{init}(p_{R_Z} - 1) - p_{R_Z})(p_{R_Z} - 1)(p_{R_{CX}} - 1)^3(4p_{R_X} - 1)e^{-\frac{t}{T_2}})dt \end{aligned} \tag{E.17}$$

$$\begin{aligned} &= \tfrac{1}{2} - p_{R_X} + \tfrac{1}{6}(p_{init} - 1)(p_{R_Z} - 1)^2(p_{R_{CX}} - 1)^2(2p_{R_X} - 1)\tfrac{\lambda_m T_1}{\lambda_m T_1 + 1} \\ &\quad + \tfrac{1}{6}(2 + p_{init}(p_{R_Z} - 1) - p_{R_Z})(p_{R_Z} - 1)(p_{R_{CX}} - 1)^3(4p_{R_X} - 1)\tfrac{\lambda_m T_2}{\lambda_m T_2 + 1}. \end{aligned} \tag{E.18}$$

Using $p_{init} = 0.006/4$, $p_{R_Z} = 0.001/3$, $p_{R_X} = 0$ and $p_{R_{CX}} = 0.005$, we obtain a post-move entanglement fidelity of

$$F^{pm}_e = \frac{3F^{pm} - 1}{2} = \frac{3(0.5 + 0.158682e^{-\frac{t}{T_1}} + 0.312165e^{-\frac{t}{T_2}}) - 1}{2} \tag{E.19}$$

$$= 0.25 + 0.238023e^{-\frac{t}{T_1}} + 0.4682475e^{-\frac{t}{T_2}}, \tag{E.20}$$

and an average post-move entanglement fidelity of

$$F^{pm}_{e,avg} = \frac{3F^{pm}_{avg} - 1}{2} = \frac{3(0.5 + 0.158682\frac{\lambda_m T_1}{\lambda_m T_1 + 1} + 0.312165\frac{\lambda_m T_2}{\lambda_m T_2 + 1}) - 1}{2} \tag{E.21}$$

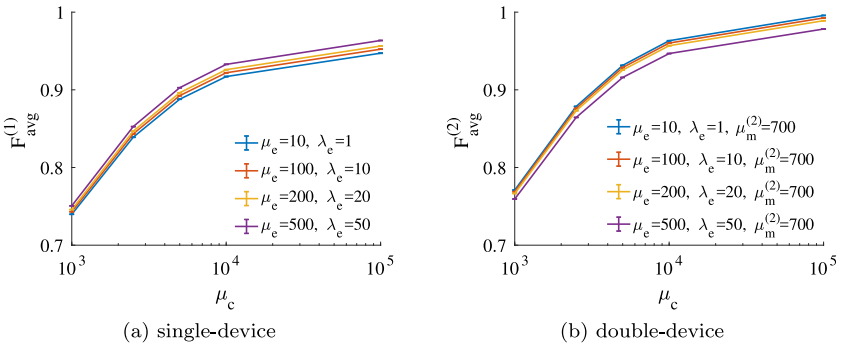
$$= 0.25 + 0.238023\frac{\lambda_m T_1}{\lambda_m T_1 + 1} + 0.4682475\frac{\lambda_m T_2}{\lambda_m T_2 + 1}. \tag{E.22}$$

From (E.22) we see that we have an upper bound of ≈ 0.956 on the average post-move fidelity.

Appendix F. Single-device and double-device average gate fidelity

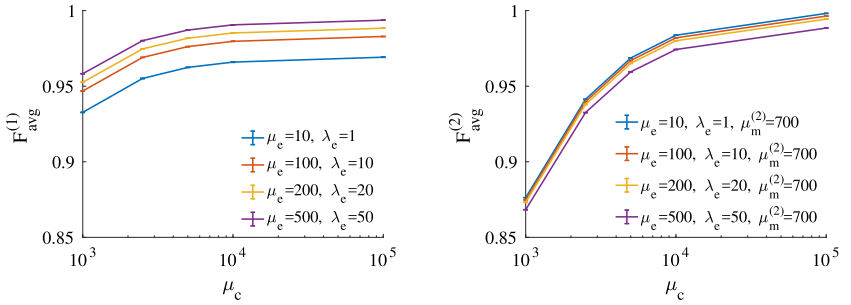
In Figs. 6 and 7, we presented differences between the SD and DD architecture average gate fidelities. In Figs. F.14 and F.15, we observe the individual average gate fidelity of each architecture design for various entanglement request and generation scenarios, as the computational job processing rate μ_c varies. Fig. F.14 corresponds to the same memory quality regime as Fig. 6: namely, one in which the memory lifetimes are equal for the two architectures. Fig. F.15 corresponds to the same memory quality regime as Fig. 7, where the memory lifetimes for the DD architecture are five times shorter than that of the SD architecture.

In both figures, we observe that as the entanglement generation rate increases, the average gate fidelity of the SD architecture increases, while the fidelity decreases for the DD architecture (in these particular examples, this holds even as the entanglement request rate scales up with the generation rate, although it may not hold in general for higher request rates). This is an expected result: recall that in the SD architecture, computational jobs wait both for entanglement generation as well as for moving requests, while in the DD architecture they only wait for moving requests. In addition, moving jobs have non-preemptive priority over computational jobs when $\mu_c < \infty$. Thus, for a fixed moving request rate, faster entanglement generation only aids computational jobs in the SD design. In contrast, in the DD design, higher entanglement generation rates lead to more moving requests, thus increasing the likelihood of computation being interrupted by these requests.



(a) single-device (b) double-device

Fig. F.14. Average gate fidelities for computational jobs for the SD architecture, (*a*), and the DD architecture, (*b*). Here, $T_1^{(1)} = T_1^{(2)} = 0.00286$s and $T_2^{(1)} = T_2^{(2)} = 0.001$s; $\mu_m^{(1)} = 1667$ Hz for all SD simulations. For all experiments, $\lambda_c = 150$ and $\lambda_m = 1000$.



(a) single-device (b) double-device

Fig. F.15. Average gate fidelities for computational jobs for the SD architecture, (*a*), and the DD architecture, (*b*). Here, $T_1^{(1)} = 10$s, $T_2^{(1)} = 0.01$s, $T_1^{(2)} = 2$s, and $T_2^{(2)} = 0.002$s; $\mu_m^{(1)} = 1667$ Hz for all SD simulations. For all experiments, $\lambda_c = 150$ and $\lambda_m = 1000$.

References

[1] J.H. Kimble, The quantum internet, Nature 453 (7198) (2008) 1023–1030, http://dx.doi.org/10.1038/nature07127.
[2] S. Wehner, D. Elkouss, R. Hanson, Quantum internet: A vision for the road ahead, Science 362 (6412) (2018) eaam9288, http://dx.doi.org/10.1126/science.aam9288.
[3] C.H. Bennett, G. Brassard, Quantum cryptography: Public key distribution and coin tossing, Theoret. Comput. Sci. 560 (2014) 7–11, http://dx.doi.org/10.1016/j.tcs.2014.05.025.
[4] A.K. Ekert, Quantum cryptography based on Bell's theorem, Phys. Rev. Lett. 67 (6) (1991) 661–663, http://dx.doi.org/10.1103/physrevlett.67.661.
[5] P. Kómár, E.M. Kessler, M. Bishof, L. Jiang, A.S. Sørensen, J. Ye, M.D. Lukin, A quantum network of clocks, Nat. Phys. 10 (8) (2014) 582–587, http://dx.doi.org/10.1038/nphys3000.
[6] A. Broadbent, J. Fitzsimons, E. Kashefi, Universal blind quantum computation, in: 2009 50th Annual IEEE Symposium on Foundations of Computer Science, IEEE, 2009, http://dx.doi.org/10.1109/focs.2009.36.
[7] L. Jiang, J.M. Taylor, A.S. Sørensen, M.D. Lukin, Distributed quantum computation based on small quantum registers, Phys. Rev. A 76 (6) (2007) http://dx.doi.org/10.1103/physreva.76.062323.
[8] V. Scarani, H. Bechmann-Pasquinucci, N.J. Cerf, M. Dušek, N. Lütkenhaus, M. Peev, The security of practical quantum key distribution, Rev. Modern Phys. 81 (3) (2009) 1301.
[9] H.J. Briegel, W. Dür, J.I. Cirac, P. Zoller, Quantum repeaters: The role of imperfect local operations in quantum communication, Phys. Rev. Lett. 81 (26) (1998) 5932–5935, http://dx.doi.org/10.1103/physrevlett.81.5932.
[10] S. Abruzzo, S. Bratzik, N.K. Bernardes, H. Kampermann, P. Van Loock, D. Bruß, Quantum repeaters and quantum key distribution: Analysis of secret-key rates, Phys. Rev. A 87 (5) (2013) 052315.
[11] S. Bratzik, S. Abruzzo, H. Kampermann, D. Bruß, Quantum repeaters and quantum key distribution: The impact of entanglement distillation on the secret key rate, Phys. Rev. A 87 (6) (2013) 062335.
[12] L. Jiang, J.M. Taylor, K. Nemoto, W.J. Munro, R. Van Meter, M.D. Lukin, Quantum repeater with encoding, Phys. Rev. A 79 (3) (2009) 032325.
[13] N. Gisin, R.T. Thew, Quantum communication technology, Electron. Lett. 46 (14) (2010) 965, http://dx.doi.org/10.1049/el.2010.1626.
[14] W.J. Munro, K. Azuma, K. Tamaki, K. Nemoto, Inside quantum repeaters, IEEE J. Sel. Top. Quantum Electron. 21 (3) (2015) 78–90, http://dx.doi.org/10.1109/jstqe.2015.2392076.
[15] C.H. Bennett, G. Brassard, S. Popescu, B. Schumacher, J.A. Smolin, W.K. Wootters, Purification of noisy entanglement and faithful teleportation via noisy channels, Phys. Rev. Lett. 76 (5) (1996) 722–725, http://dx.doi.org/10.1103/physrevlett.76.722.
[16] D. Deutsch, A. Ekert, R. Jozsa, C. Macchiavello, S. Popescu, A. Sanpera, Quantum privacy amplification and the security of quantum cryptography over noisy channels, Phys. Rev. Lett. 77 (13) (1996) 2818–2821, http://dx.doi.org/10.1103/physrevlett.77.2818.
[17] W. Dür, H.J. Briegel, Entanglement purification and quantum error correction, Rep. Progr. Phys. 70 (8) (2007) 1381.
[18] A. Dahlberg, M. Skrzypczyk, T. Coopmans, L. Wubben, F. Rozpędek, M. Pompili, A. Stolk, P. Pawełczak, R. Knegjens, J. de Oliveira Filho, R. Hanson, S. Wehner, A link layer protocol for quantum networks, in: Proceedings of the ACM Special Interest Group on Data Communication, ACM, 2019, http://dx.doi.org/10.1145/3341302.3342070.

[19] M. Pompili, S.L. Hermans, S. Baier, H.K. Beukers, P.C. Humphreys, R.N. Schouten, R.F. Vermeulen, M.J. Tiggelman, L. dos Santos Martins, B. Dirkse, et al., Realization of a multinode quantum network of remote solid-state qubits, Science 372 (6539) (2021) 259–264.
[20] V. Krutyanskiy, M. Meraner, J. Schupp, V. Krcmarsky, H. Hainzer, B.P. Lanyon, Light-matter entanglement over 50 km of optical fibre, Npj Quantum Inf. 5 (1) (2019) http://dx.doi.org/10.1038/s41534-019-0186-3.
[21] N. Sangouard, R. Dubessy, C. Simon, Quantum repeaters based on single trapped ions, Phys. Rev. A 79 (4) (2009) 042340.
[22] A.D. Pfister, M. Salz, M. Hettrich, U.G. Poschinger, F. Schmidt-Kaler, A quantum repeater node with trapped ions: a realistic case example, Appl. Phys. B 122 (4) (2016) 89.
[23] N. Sangouard, C. Simon, H. de Riedmatten, N. Gisin, Quantum repeaters based on atomic ensembles and linear optics, Rev. Modern Phys. 83 (1) (2011) 33–80, http://dx.doi.org/10.1103/revmodphys.83.33.
[24] B. Hensen, H. Bernien, A.E. Dréau, A. Reiserer, N. Kalb, M.S. Blok, J. Ruitenberg, R.F.L. Vermeulen, R.N. Schouten, C. Abellán, W. Amaya, V. Pruneri, M.W. Mitchell, M. Markham, D.J. Twitchen, D. Elkouss, S. Wehner, T.H. Taminiau, R. Hanson, Loophole-free Bell inequality violation using electron spins separated by 1.3 kilometres, Nature 526 (7575) (2015) 682–686, http://dx.doi.org/10.1038/nature15759.
[25] M.H. Abobeih, J. Cramer, M.A. Bakker, N. Kalb, M. Markham, D.J. Twitchen, T.H. Taminiau, One-second coherence for a single electron spin coupled to a multi-qubit nuclear-spin environment, Nature Commun. 9 (1) (2018) http://dx.doi.org/10.1038/s41467-018-04916-z.
[26] P.C. Humphreys, N. Kalb, J.P.J. Morits, R.N. Schouten, R.F.L. Vermeulen, D.J. Twitchen, M. Markham, R. Hanson, Deterministic delivery of remote entanglement on a quantum network, Nature 558 (7709) (2018) 268–273, http://dx.doi.org/10.1038/s41586-018-0200-5.
[27] W. Pfaff, B.J. Hensen, H. Bernien, S.B. van Dam, M.S. Blok, T.H. Taminiau, M.J. Tiggelman, R.N. Schouten, M. Markham, D.J. Twitchen, R. Hanson, Unconditional quantum teleportation between distant solid-state quantum bits, Science 345 (6196) (2014) 532–535, http://dx.doi.org/10.1126/science.1253512.
[28] T. Coopmans, R. Knegjens, A. Dahlberg, D. Maier, L. Nijsten, J. Oliveira, M. Papendrecht, J. Rabbie, F. Rozpędek, M. Skrzypczyk, et al., NetSquid, a discrete-event simulation platform for quantum networks, 2020, arXiv preprint arXiv:2010.12535.
[29] A. Reiserer, N. Kalb, M.S. Blok, K.J.M. van Bemmelen, T.H. Taminiau, R. Hanson, D.J. Twitchen, M. Markham, Robust quantum-network memory using decoherence-protected subspaces of nuclear spins, Phys. Rev. X 6 (2) (2016) http://dx.doi.org/10.1103/physrevx.6.021040.
[30] E. Kawakami, T. Jullien, P. Scarlino, D.R. Ward, D.E. Savage, M.G. Lagally, V.V. Dobrovitski, M. Friesen, S.N. Coppersmith, M.A. Eriksson, L.M.K. Vandersypen, Gate fidelity and coherence of an electron spin in an Si/SiGe quantum dot with micromagnet, Proc. Natl. Acad. Sci. 113 (42) (2016) 11738–11743, http://dx.doi.org/10.1073/pnas.1603251113.
[31] J.M. Nichol, L.A. Orona, S.P. Harvey, S. Fallahi, G.C. Gardner, M.J. Manfra, A. Yacoby, High-fidelity entangling gate for double-quantum-dot spin qubits, Npj Quantum Inf. 3 (1) (2017) http://dx.doi.org/10.1038/s41534-016-0003-1.
[32] S. Guha, H. Krovi, C.A. Fuchs, Z. Dutton, J.A. Slater, C. Simon, W. Tittel, Rate-loss analysis of an efficient quantum repeater architecture, Phys. Rev. A 92 (2) (2015) http://dx.doi.org/10.1103/physreva.92.022357.
[33] E. Shchukin, F. Schmidt, P. van Loock, Waiting time in quantum repeaters with probabilistic entanglement swapping, Phys. Rev. A 100 (3) (2019) http://dx.doi.org/10.1103/physreva.100.032322.
[34] S. Brand, T. Coopmans, D. Elkouss, Efficient computation of the waiting time and fidelity in quantum repeater chains, in: OSA Quantum 2.0 Conference, OSA, 2020, http://dx.doi.org/10.1364/quantum.2020.qth7a.11.
[35] B. Li, T. Coopmans, D. Elkouss, Efficient optimization of cut-offs in quantum repeater chains, in: 2020 IEEE International Conference on Quantum Computing and Engineering, QCE, IEEE, 2020, http://dx.doi.org/10.1109/qce49297.2020.00029.
[36] G. Vardoyan, S. Guha, P. Nain, D. Towsley, On the stochastic analysis of a quantum entanglement distribution switch, IEEE Trans. Quantum Eng. 2 (2021) 1–16, http://dx.doi.org/10.1109/tqe.2021.3058058.
[37] G. Vardoyan, S. Guha, P. Nain, D. Towsley, On the exact analysis of an idealized quantum switch, Perform. Eval. 144 (2020) 102141, http://dx.doi.org/10.1016/j.peva.2020.102141.
[38] G. Vardoyan, S. Guha, P. Nain, D. Towsley, On the capacity region of bipartite and tripartite entanglement switching, in: Performance 2020-38th IFIP International Symposium on Computer Performance, Modeling, Measurements and Evaluation, 2020, pp. 1–6.
[39] M.A. Nielsen, I.L. Chuang, Quantum Computation and Quantum Information, Cambridge University Press, http://dx.doi.org/10.1017/cbo9780511976667.016.
[40] D.G. Tempel, A. Aspuru-Guzik, Relaxation and dephasing in open quantum systems time-dependent density functional theory: Properties of exact functionals from an exactly-solvable model system, Chem. Phys. 391 (1) (2011) 130–142.
[41] C.D. Bruzewicz, J. Chiaverini, R. McConnell, J.M. Sage, Trapped-ion quantum computing: Progress and challenges, Appl. Phys. Rev. 6 (2) (2019) 021314.
[42] P. Jobez, N. Timoney, C. Laplane, J. Etesse, A. Ferrier, P. Goldner, N. Gisin, M. Afzelius, Towards highly multimode optical quantum memory for quantum repeaters, Phys. Rev. A 93 (3) (2016) 032327.
[43] A. Gilchrist, N.K. Langford, M.A. Nielsen, Distance measures to compare real and ideal quantum processes, Phys. Rev. A 71 (6) (2005) 062310.
[44] R. Van Meter, Quantum Networking, John Wiley & Sons, 2014.
[45] M.A. Nielsen, A simple formula for the average gate fidelity of a quantum dynamical operation, Phys. Lett. A 303 (4) (2002) 249–252.
[46] B. Schumacher, Sending entanglement through noisy quantum channels, Phys. Rev. A 54 (4) (1996) 2614–2628, http://dx.doi.org/10.1103/physreva.54.2614.
[47] G. Latouche, V. Ramaswami, Introduction to Matrix Analytic Methods in Stochastic Modeling, SIAM, 1999.
[48] M. Pinsky, S. Karlin, An Introduction to Stochastic Modeling, Academic Press, 2010.
[49] M.D. Bowdrey, D.K. Oi, A.J. Short, K. Banaszek, J.A. Jones, Fidelity of single qubit maps, Phys. Lett. A 294 (5–6) (2002) 258–260.
[50] M. Ruf, N.H. Wan, H. Choi, D. Englund, R. Hanson, Quantum networks based on color centers in diamond, 2021, arXiv preprint arXiv:2105.04341.
[51] D. Gottesman, H.-K. Lo, Proof of security of quantum key distribution with two-way classical communications, IEEE Trans. Inf. Theory 49 (2) (2003) 457–475, http://dx.doi.org/10.1109/tit.2002.807289.
[52] M.F. Neuts, Matrix-Geometric Solutions in Stochastic Models: An Algorithmic Approach, Courier Corporation, 1994.
[53] A. Marin, S.R. Bulò, Explicit solutions for queues with hypo-exponential service time and applications to product-form analysis, in: Proceedings of the 5th International ICST Conference on Performance Evaluation Methodologies and Tools, 2011, pp. 166–175.
[54] C. Bradley, J. Randall, M.H. Abobeih, R. Berrevoets, M. Degen, M. Bakker, M. Markham, D. Twitchen, T.H. Taminiau, A ten-qubit solid-state spin register with quantum memory up to one minute, Phys. Rev. X 9 (3) (2019) 031045.
[55] N. Kalb, A.A. Reiserer, P.C. Humphreys, J.J. Bakermans, S.J. Kamerling, N.H. Nickerson, S.C. Benjamin, D.J. Twitchen, M. Markham, R. Hanson, Entanglement distillation between solid-state quantum network nodes, Science 356 (6341) (2017) 928–932.
[56] F. Rozpędek, R. Yehia, K. Goodenough, M. Ruf, P.C. Humphreys, R. Hanson, S. Wehner, D. Elkouss, Near-term quantum-repeater experiments with nitrogen-vacancy centers: Overcoming the limitations of direct transmission, Phys. Rev. A 99 (5) (2019) 052330.
[57] N. Kalb, P.C. Humphreys, J.J. Slim, R. Hanson, Dephasing mechanisms of diamond-based nuclear-spin memories for quantum networks, Phys. Rev. A 97 (6) (2018) 062330.
[58] T.H. Taminiau, J. Cramer, T. van der Sar, V.V. Dobrovitski, R. Hanson, Universal control and error correction in multi-qubit spin registers in diamond, Nature Nanotechnol. 9 (3) (2014) 171.
[59] M.M. Wolf, Quantum Channels & Operations: Guided Tour, Vol. 5, Citeseer, 2012, Lecture notes available at http://www-m5.ma.tum.de/foswiki/pubM.
[60] J.M. Renes, R. Blume-Kohout, A.J. Scott, C.M. Caves, Symmetric informationally complete quantum measurements, J. Math. Phys. 45 (6) (2004) 2171–2180.